

全国高等职业教育计算机类规划教材·工作过程系统化教程系列

ASP.NET 3.5 项目开发实战

宋海兰 李 航 沙继东 主编
姜惠民 主审

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书以电子商务网上购书信息管理系统、企业新闻发布信息管理系统、企业在线客服管理系统及商业网站流量分析管理系统4个企业级项目开发为例,从软件工程的角度出发,以软件开发的工作过程为主线,系统、全面地介绍程序开发流程;从项目背景、需求分析、系统架构设计、子系统设计、界面设计、数据库设计、网站开发到网站的生成与发布,每一个过程都有详细的介绍。

本书提供的所有源代码都经过精心调试,在 Windows XP 和 Windows Server 2003 操作系统下全部通过,保证能够正常运行。读者也可以对案例源代码和数据库进行二次开发,以缩短开发系统所需要的时间。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

ASP.NET 3.5 项目开发实战 / 宋海兰, 李航, 沙继东主编. —北京: 电子工业出版社, 2009.8

全国高等职业教育计算机类规划教材·工作过程系统化教程系列

ISBN 978-7-121-08961-9

I.A… II. ①宋…②李…③沙… III. 主页制作—程序设计—高等学校: 技术学校—教材 IV.TP393.092

中国版本图书馆 CIP 数据核字(2009)第 086186 号

策划编辑: 程超群

责任编辑: 徐 萍

印 刷:

装 订:

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×1 092 1/16 印张: 25.5 字数: 648 千字

印 次: 2009 年 8 月第 1 次印刷

印 数: 4 000 册 定价: 39.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

前 言

本书以 Visual Studio 2008 作为开发环境，将 ASP.NET 3.5 的各项技术融入到典型业务类型的 Web 应用程序中进行讲解；以 4 个企业级项目开发实例的工作过程为主线，并由软件企业从业人员全程参与，使用 UML 建模语言对系统设计、子系统设计、界面设计、数据库设计加以介绍；系统实现采用典型的分层结构，并应用最新的设计模式；测试及发布遵循规范；项目中所涉及的每一个知识点都根据实际开发的需要来讲解，步骤详细、可操作性强，并极具代表性。

本书具有以下特点。

(1) 技术最新：以 Visual Studio 2008 作为开发环境，详细介绍 ASP.NET 的各项技术。

(2) 实战性强：介绍 4 个典型业务类型 Web 应用程序的开发，以工作过程为主线，并以工作任务为核心，采用任务驱动方式。

(3) 不但可以为具有一定基础的程序员提供参考，而且能够满足软件技术专业“基于工作过程系统化课程”建设的学习情境设计与开发的需要，适合选作生产性实训的教材。

(4) 项目中所涉及的每一个知识点都根据实际开发的需要来讲解，融理论与实践为一体。

本书包括如下内容。

第 1 章 电子商务网上购书信息管理系统，主要内容包括系统分析与设计、图书信息浏览、图书检索、购物车与订单管理、图书类别及图书管理等。

第 2 章 企业新闻发布信息管理系统，主要内容包括系统分析与设计、新闻浏览、新闻评论、新闻类别、新闻添加与维护、新闻审核等。

第 3 章 企业在线客服信息管理系统（AJAX 技术应用），主要内容包括系统分析与设计、前台（在线/离线）消息发送、浏览、获取客服列表实现及后台客服管理、后台消息管理、客服消息发送管理实现等。

第 4 章 商业网站流量统计分析系统，主要内容包括系统分析与设计、站点流量统计、栏目流量统计、IP 流量统计等。

本书由宋海兰、李航、沙继东主编，姜惠民主审，参与编写的其他人员还有王依楠、田晶、李亚楠、程志凯等，其中李亚楠、程志凯来自于软件企业。读者对本书有任何建议，可发 E-mail 至 hellensong2009@126.com。因编者水平有限，若有错漏之处，望来函告知。

编 者
2009 年 4 月

目 录

第 1 章 电子商务网上购书信息管理系统.....	(1)
1.1 用户需求分析与处理	(1)
1.1.1 任务名称：用户需求分析与处理	(1)
1.1.2 任务描述	(1)
1.1.3 任务分析	(1)
1.1.4 收集用户需求	(1)
1.1.5 需求分析人员分析用户的需求	(4)
1.1.6 任务小结	(11)
1.1.7 练习题	(12)
1.2 项目计划安排	(12)
1.2.1 任务名称：项目计划安排	(13)
1.2.2 任务描述	(13)
1.2.3 任务分析	(13)
1.2.4 项目概述	(14)
1.2.5 主要参加人员	(14)
1.2.6 应交付成果	(14)
1.2.7 验收标准	(15)
1.2.8 完成项目的最迟期限	(15)
1.2.9 实施计划	(15)
1.2.10 系统运行软硬件环境	(16)
1.2.11 任务小结	(17)
1.2.12 练习题	(17)
1.3 系统设计	(17)
1.3.1 任务名称：系统设计	(17)
1.3.2 任务描述	(17)
1.3.3 任务分析	(18)
1.3.4 层图（逻辑视图）	(20)
1.3.5 包图（开发视图）	(20)
1.3.6 部署图（物理视图）	(21)
1.3.7 选择技术	(21)
1.3.8 安全策略	(21)
1.3.9 并发策略	(22)
1.3.10 任务小结	(22)
1.3.11 练习题	(22)
1.4 子系统设计	(23)
1.4.1 任务名称：子系统设计	(23)

1.4.2	任务描述	(23)
1.4.3	任务分析	(23)
1.4.4	类的列表	(24)
1.4.5	类的规格说明示例	(25)
1.4.6	用例具体实现示例	(26)
1.4.7	系统用户界面总览	(28)
1.4.8	数据库设计	(31)
1.4.9	任务小结	(35)
1.4.10	练习题	(35)
1.5	开发前期的解决方案构建	(36)
1.5.1	任务名称: 开发前期的解决方案构建	(36)
1.5.2	任务描述	(36)
1.5.3	任务分析	(36)
1.5.4	创建解决方案	(36)
1.5.5	在类库 Model 中创建业务实体类	(38)
1.5.6	在类库 Common 中创建公共类数据库连接类 DataBase	(49)
1.5.7	设计解决方案中网站 Web 的母版页	(54)
1.5.8	MasterPage.master.cs 代码实现	(57)
1.5.9	为网站 Web 设置主题	(66)
1.5.10	创建站点地图	(67)
1.5.11	任务小结	(67)
1.5.12	练习题	(67)
1.6	前台图书信息浏览、检索实现	(68)
1.6.1	任务名称: 前台图书信息浏览、检索实现	(68)
1.6.2	任务描述	(68)
1.6.3	任务分析	(68)
1.6.4	首页分栏目显示图书信息	(68)
1.6.5	按栏目或者图书类别显示图书概要信息	(74)
1.6.6	图书详情信息查看	(78)
1.6.7	复合条件图书检索	(82)
1.6.8	任务小结	(91)
1.6.9	练习题	(91)
1.7	前台用户的注册、修改个人资料实现	(91)
1.7.1	任务名称: 前台用户的注册、修改个人资料实现	(91)
1.7.2	任务描述	(91)
1.7.3	任务分析	(91)
1.7.4	创建或管理角色及设置角色的访问权限	(92)
1.7.5	会员注册	(94)
1.7.6	修改会员个人资料	(95)
1.7.7	任务小结	(98)

1.7.8 练习题..... (98)

1.8 前台购物车管理..... (98)

1.8.1 任务名称：前台购物车管理..... (98)

1.8.2 任务描述..... (98)

1.8.3 任务分析..... (99)

1.8.4 Model 层：购物车实体类 CartInfo 类的实现..... (99)

1.8.5 DAL 层：购物车数据访问类 CartAccess 类的实现..... (101)

1.8.6 BLL 层：购物车业务逻辑类 CartManager 类的实现..... (105)

1.8.7 购物车表示层代码的实现..... (110)

1.8.8 任务小结..... (118)

1.8.9 练习题..... (118)

1.9 前台订单管理实现..... (119)

1.9.1 任务名称：前台订单管理实现..... (119)

1.9.2 任务描述..... (119)

1.9.3 任务分析..... (119)

1.9.4 在 Model 类库中创建 OrdersInfo、LineItemInfo 等业务实体类..... (119)

1.9.5 在数据库中创建存储过程与触发器..... (119)

1.9.6 在 DAL 类库中创建 OrderAccess 类..... (120)

1.9.7 在 BLL 类库中创建 OrderManager 类..... (123)

1.9.8 表示层 CheckOut.aspx 页面的实现..... (124)

1.9.9 会员查看自己的订单实现..... (130)

1.9.10 任务小结..... (133)

1.9.11 练习题..... (133)

1.10 后台图书类别管理实现..... (133)

1.10.1 任务名称：实现图书类别管理..... (133)

1.10.2 任务描述..... (133)

1.10.3 任务分析..... (133)

1.10.4 在 Model 类库中创建目录实体类 CategoryInfo..... (133)

1.10.5 DAL 层：目录添加数据访问类 CategoryAccess 类的实现..... (134)

1.10.6 BLL 层：目录添加管理业务逻辑类 CategoryManager 类的实现..... (137)

1.10.7 目录添加管理表示层代码实现..... (137)

1.10.8 目录管理功能实现..... (144)

1.10.9 BLL 层：CategoryManager 类完善目录删除功能..... (146)

1.10.10 目录删除表示层设计..... (147)

1.10.11 目录修改表示层 EditCategory.aspx 代码实现..... (149)

1.10.12 目录管理页 CategoryManager.aspx..... (152)

1.10.13 任务小结..... (154)

1.10.14 练习题..... (155)

1.11 后台图书信息管理..... (156)

1.11.1 任务名称：后台图书信息管理..... (156)

1.11.2	任务描述	(156)
1.11.3	任务分析	(156)
1.11.4	Model 层: 图书详情 BookBriefInfo 类实现	(156)
1.11.5	DAL 层: 图书详情访问类 BookBriefAccess 类的实现	(158)
1.11.6	BLL 层: 图书详情 BookBriefManager 类的实现	(162)
1.11.7	图书信息管理表示层代码实现	(163)
1.11.8	任务小结	(170)
1.11.9	练习题	(170)
1.12	生成及发布网站	(170)
1.12.1	任务名称: 生成及发布网站	(171)
1.12.2	任务描述	(171)
1.12.3	任务分析	(171)
1.12.4	生成网站及发布网站	(171)
1.12.5	任务小结	(175)
1.12.6	练习题	(175)
第 2 章	企业新闻发布信息管理系统	(176)
2.1	用户需求的分析与处理	(176)
2.1.1	任务名称: 用户需求的分析与处理	(176)
2.1.2	任务描述	(176)
2.1.3	任务分析	(176)
2.1.4	收集用户需求	(177)
2.1.5	分析用户的需求	(179)
2.1.6	任务小结	(180)
2.1.7	练习题	(180)
2.2	项目计划安排	(180)
2.2.1	任务名称: 项目计划安排	(180)
2.2.2	任务描述	(180)
2.2.3	任务分析	(180)
2.2.4	项目计划	(181)
2.2.5	任务小结	(182)
2.2.6	练习题	(182)
2.3	系统设计	(182)
2.3.1	任务名称: 系统架构设计	(182)
2.3.2	任务描述	(182)
2.3.3	任务分析	(183)
2.3.4	架构重点及模式	(183)
2.3.5	选择技术	(184)
2.3.6	安全策略	(184)
2.3.7	任务小结	(184)
2.3.8	练习题	(185)

2.4	子系统设计	(185)
2.4.1	任务名称：子系统设计	(185)
2.4.2	任务描述	(185)
2.4.3	任务分析	(185)
2.4.4	类的列表	(185)
2.4.5	数据库设计	(186)
2.4.6	任务小结	(188)
2.4.7	练习题	(188)
2.5	新闻信息显示与检索实现	(188)
2.5.1	任务名称：新闻信息显示与检索实现	(189)
2.5.2	任务描述	(189)
2.5.3	任务分析	(189)
2.5.4	Model 层：实体类实现	(190)
2.5.5	DAL 层：数据访问类实现	(193)
2.5.6	BLL 层：业务逻辑类实现	(204)
2.5.7	任务小结	(208)
2.5.8	练习题	(208)
2.6	新闻评论实现	(209)
2.6.1	任务名称：新闻评论实现	(209)
2.6.2	任务描述	(209)
2.6.3	任务分析	(209)
2.6.4	Model 层：CommentsInfo 类实现	(209)
2.6.5	DAL 层：CommentsAccess 类实现	(211)
2.6.6	BLL 层：CommentsLogic 类实现	(213)
2.6.7	任务小结	(214)
2.7	后台用户管理实现	(214)
2.7.1	任务名称：后台用户管理实现	(214)
2.7.2	任务描述	(214)
2.7.3	任务分析	(215)
2.7.4	Model 层：UserInfo 类实现	(215)
2.7.5	DAL 层：UserAccess 类实现	(216)
2.7.6	BLL 层：UserLogic 类实现	(218)
2.7.7	任务小结	(220)
2.8	母版页设计	(220)
2.8.1	任务名称：母版页设计	(220)
2.8.2	任务描述	(220)
2.8.3	任务分析	(221)
2.8.4	任务完成	(221)
2.8.5	任务小结	(231)
2.8.6	练习题	(231)

2.9 网站前台的实现..... (231)

2.9.1 任务名称：网站前台的实现..... (232)

2.9.2 任务描述..... (232)

2.9.3 任务分析..... (232)

2.9.4 网站前台首页 Default.aspx..... (232)

2.9.5 新闻栏目页面 BigTypeNews.aspx..... (237)

2.9.6 新闻内容浏览及评论页面 ListView.aspx..... (241)

2.9.7 新闻全部评论浏览页面 MoreComments.aspx..... (251)

2.9.8 全部新闻页面 AllNews.aspx..... (254)

2.9.9 新闻搜索页面 Search.aspx..... (257)

2.9.10 用户注册界面 UserReg.aspx..... (260)

2.9.11 用户发布新闻信息界面 UserAddNews.aspx..... (263)

2.9.12 个人管理信息页面 UserCenter.aspx..... (266)

2.9.13 任务小结..... (270)

2.10 网站后台的实现..... (271)

2.10.1 任务名称：网站后台的实现..... (271)

2.10.2 任务描述..... (271)

2.10.3 任务分析..... (271)

2.10.4 网站后台登录页面 Admin_Login.aspx..... (271)

2.10.5 网站后台管理首页 Admin_Index.aspx..... (273)

2.10.6 网站后台管理现有新闻页面 Admin_NewsList.aspx..... (276)

2.10.7 网站后台修改新闻页面 Admin_EditNews.aspx..... (281)

2.10.8 网站后台删除新闻页面 Admin_DeleteNews.aspx..... (283)

2.10.9 网站后台发布新闻页面 Admin_AddNews.aspx..... (284)

2.10.10 网站后台审核新闻页面 Admin_CheckNews.aspx..... (287)

2.10.11 网站后台管理审核功能页面 CheckNews.aspx..... (291)

2.10.12 网站后台管理新闻评论页面 Admin_Comments.aspx..... (292)

2.10.13 网站后台管理新闻类别页面 Admin_BigClass.aspx..... (296)

2.10.14 网站后台管理修改新闻类别页面 Admin_EditBig.aspx..... (299)

2.10.15 网站后台管理删除新闻类别页面 Admin_DeleteBig.aspx..... (301)

2.10.16 网站后台管理系统用户页面 Admin_AllUsers.aspx..... (303)

2.10.17 网站后台修改用户信息页面 Admin_EditUser.aspx..... (307)

2.10.18 网站后台管理删除用户信息页面 Admin_DeleteUser.aspx..... (309)

2.10.19 任务小结..... (310)

2.11 系统的测试..... (310)

2.11.1 任务名称：系统的测试..... (311)

2.11.2 任务描述..... (311)

2.11.3 任务分析..... (311)

2.11.4 系统的测试..... (311)

2.11.5 NUnit 测试工具..... (312)

2.11.6	任务小结	(313)
2.11.7	练习题	(313)
第3章	企业在线客服信息管理系统（AJAX 技术应用）	(314)
3.1	用户需求的分析与处理	(314)
3.1.1	任务名称：用户需求的分析与处理	(314)
3.1.2	任务描述	(314)
3.1.3	任务分析：需求分析人员分析用户的需求	(315)
3.1.4	需求建模	(316)
3.1.5	撰写规格说明书	(317)
3.1.6	任务小结	(318)
3.1.7	练习题	(318)
3.2	项目计划安排	(319)
3.2.1	任务名称：项目计划安排	(319)
3.2.2	任务描述	(319)
3.2.3	任务分析	(319)
3.2.4	创建甘特图	(319)
3.2.5	实现项目的资源	(320)
3.2.6	编写前期的项目计划表	(321)
3.2.7	任务小结	(321)
3.2.8	练习题	(321)
3.3	系统架构设计	(321)
3.3.1	任务名称：系统架构设计	(321)
3.3.2	任务描述	(321)
3.3.3	任务分析	(321)
3.3.4	任务小结	(322)
3.3.5	练习题	(322)
3.4	模块的详细设计	(322)
3.4.1	任务名称：模块的详细设计	(322)
3.4.2	任务描述	(322)
3.4.3	任务分析	(323)
3.4.4	类的列表及规格说明	(323)
3.4.5	用图例实现设计	(323)
3.4.6	用例的基本事件流、扩展事件流、异常事件流	(324)
3.4.7	用例的顺序图与活动图	(324)
3.4.8	任务小结	(326)
3.4.9	练习题	(326)
3.5	数据库设计	(326)
3.5.1	任务名称：数据库设计	(326)
3.5.2	任务描述	(326)
3.5.3	任务分析	(326)

3.5.4	生成数据库.....	(327)
3.5.5	任务小结	(329)
3.5.6	练习题	(329)
3.6	界面设计	(329)
3.6.1	任务名称：界面设计	(330)
3.6.2	任务描述	(330)
3.6.3	任务分析	(330)
3.6.4	前期准备	(330)
3.6.5	相关技能与知识	(333)
3.6.6	任务小结	(333)
3.6.7	练习题	(333)
3.7	前台用户的登录、注册、个人设置、修改个人资料实现.....	(334)
3.7.1	任务名称：前台用户的登录、注册、个人设置、修改个人资料实现.....	(334)
3.7.2	任务描述	(334)
3.7.3	任务分析	(334)
3.7.4	Model 层：用户实体类 User 类的实现	(334)
3.7.5	DAL 层：数据访问类 DataBase 类的实现	(336)
3.7.6	BLL 层：业务逻辑类的实现	(340)
3.7.7	登录页面表示层代码的实现.....	(341)
3.7.8	相关技能与知识点	(343)
3.7.9	任务小结	(343)
3.7.10	练习题	(344)
3.8	前台（在线/离线）消息发送、浏览、获取客服列表实现.....	(344)
3.8.1	任务名称：前台（在线/离线）消息发送、浏览、获取客服列表实现.....	(344)
3.8.2	任务描述	(344)
3.8.3	任务分析	(344)
3.8.4	Model 层：消息实体类 Message 类的实现	(345)
3.8.5	DAL 层：数据访问类 MessageDAL 类的实现.....	(346)
3.8.6	BLL 层：业务逻辑类的实现	(348)
3.8.7	消息发送、消息浏览表示层代码的实现.....	(349)
3.8.8	相关技能与知识	(352)
3.8.9	任务小结	(353)
3.8.10	练习题	(353)
3.9	后台客服管理.....	(353)
3.9.1	任务名称：后台客服管理.....	(353)
3.9.2	任务描述	(353)
3.9.3	任务分析	(354)
3.9.4	Model 层：实体类的实现	(354)
3.9.5	DAL 层：数据访问类的实现.....	(354)
3.9.6	BLL 层：业务逻辑类的实现	(355)

3.9.7	客服管理表示层代码的实现	(355)
3.9.8	任务小结	(357)
3.9.9	练习题	(357)
3.10	后台消息管理实现	(357)
3.10.1	任务名称: 后台消息管理实现	(357)
3.10.2	任务描述	(357)
3.10.3	任务分析	(357)
3.10.4	Model 层: 实体类的实现	(357)
3.10.5	DAL 层: 数据访问类的实现	(357)
3.10.6	BLL 层: 业务逻辑类的实现	(358)
3.10.7	消息管理表示层代码的实现	(358)
3.10.8	任务小结	(359)
3.10.9	练习题	(360)
3.11	客服消息发送管理实现	(360)
3.11.1	任务名称: 客服消息发送管理实现	(360)
3.11.2	任务描述	(360)
3.11.3	任务分析	(360)
3.11.4	Model 层: 实体类的实现	(361)
3.11.5	DAL 层: 数据访问类方法的实现	(361)
3.11.6	消息发送、浏览表示层代码的实现	(361)
3.11.7	相关技能与知识	(364)
3.11.8	任务小结	(364)
3.11.9	练习题	(364)
第 4 章	商业网站流量统计分析系统	(365)
4.1	系统设计	(365)
4.1.1	任务名称: 系统设计	(365)
4.1.2	任务描述	(365)
4.1.3	任务分析	(365)
4.1.4	任务准备	(365)
4.1.5	创建商业网站流量分析系统网站母版页	(367)
4.1.6	设计显示当前日期和时间的 Web 自定义控件 DisplayTime.ascx	(375)
4.1.7	设计数据访问层的数据访问类 DataBase.cs	(376)
4.1.8	设计显示当月访问量的 Web 用户控件 DisplayMonthCount.ascx	(379)
4.1.9	设计显示当日访问量 Web 用户控件 getDayCount.ascx	(380)
4.2	系统实现	(380)
4.2.1	site 站点流量统计	(380)
4.2.2	func 栏目流量统计	(384)
4.2.3	IP 流量统计	(387)
4.2.4	相关技能与要点	(390)
4.2.5	任务小结	(390)
4.2.6	练习题	(390)
参考文献	(391)

第 1 章 电子商务网上购书信息管理系统

1.1 用户需求分析与处理

学习目标

■ 理解需求阶段的目标

■ 给业务上下文和系统功能建模

■ 在完整的用例模型中记录系统需求

■ 建议学时：8 学时

1.1.1 任务名称：用户需求分析与处理

1.1.2 任务描述

依据电子商务的特点与基本流程及用户需求调查报告，通过理解需求阶段的目标，为业务上下文和系统功能建模，在完整的用例模型中记录系统需求，完成需求模型报告，最后依据需求模型报告进行产品需求规格说明书的撰写。

1.1.3 任务分析

需求分析人员要按“初始、细化、构造与移交四步走”的路线，通过“以目标为基础、以用例为中心的三次迭代式需求分析”的过程来完成对用户需求的分析。

（1）第一次迭代（初始）：学会进行项目目标分解、进行项目目标可行性研究分析，构造提交项目目标模型，形成项目大纲。

（2）第二次迭代（细化）：学会用例图建模，进行客户需求分析，构造提供软件功能模型，形成客户需求文档。

（3）第三次迭代（构造）：学会对用例进行“三位”一体的描述方式，分析软件用例的动态行为，构造提交用例的业务流程图、实体类图、原型图，形成产品需求说明书。

（4）需求验证（移交）：学会从需求类型与属性角度评估需求的质量，移交产品需求说明书。

1.1.4 收集用户需求

1. 项目的背景及意义

网络经济是用现代信息技术和网络技术，依靠形成的互联网网络进行商务活动的，它集金融电子化、管理信息化、办公室自动化于一体，与传统商务模式相比较，具有无可比拟的运作优势：

（1）运营成本低；

- (2) 用户范围广；
- (3) 商务开放性好；
- (4) 无时空限制；
- (5) 多媒体手段表现力强；
- (6) 以顾客为中心，最大限度地满足顾客的个性化需求；
- (7) 提升企业形象；

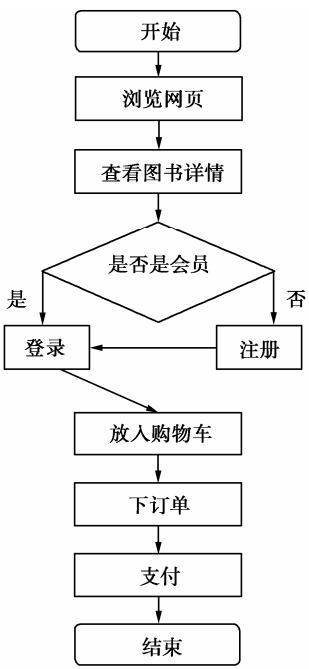


图 1-1 网上购书业务流程

以上可参考网上购物视频资源 <http://you.video.sina.com.cn/b/14280103-1395124393.html>。

3. 用户需求调查问卷

用户需求调查问卷如下所示。

用户需求调查问卷

作 者	
日 期	

提示：调查问卷被分发给用户填写，可以根据预期的调查内容剪裁。

填写人姓名：

单位：

职务：

职责：

需求名称	描 述	需求类型
需求 1		
需求 2		

需求名称	描述	需求类型
需求 3		
需求 4		
需求 5		
备注		

调查问卷附录

需求类型

- 功能性
- 可用性
- 可靠性
- 性能
- 可支持性
- 设计需求
- 实施需求
- 接口需求
- 物理需求

功能性

功能性需求包括特性功能和安全性。

可用性

可用性需求包含如下子类别：

- 人员因素；
- 美观；
- 用户界面的一致性；
- 联机帮助和环境相关帮助；
- 向导和代理；
- 用户文档和培训材料。

可靠性

需要考虑的可靠性需求有：故障的频率/严重性、可恢复性、可预见性、准确性和平均故障间隔时间（MTBF）。

性能

性能需求可对功能性需求强加条件。例如，对于一个给定行为，它可以对以下项规定性能参数：速度、效率、可用性、准确性、吞吐量、响应时间、恢复时间或资源用途。

可支持性

可支持性需求包括：可测试性、可扩展性、可适应性、可维护性、兼容性、可配置性、可服务性、可安装性，或是否可本地化（国际化）。

设计需求

设计需求常称为设计约束，它规定或约束了系统的设计。

实施需求

实施需求规定或约束了系统的编码或构建。例如，所需标准、实施语言、数据库完整性策略、资源限制和操作环境。

接口需求

接口需求规定了系统必须与之交互操作的外部项，或对这种交互操作所使用的格式、时间或其他因素的约束。

物理需求

物理需求规定了系统必须具备的物理特征，如材质、形状、尺寸和质量。这种需求类型可用来代表硬件要求，如物理网络配置需求。

4. 用户需求

(1) 顾客可以匿名浏览书的目录和所有书籍的概要信息及详情信息，但需登录具有会员资格才能购买图书。

(2) 顾客可以按图书类别浏览图书信息。

(3) 提供图书的快速检索功能及高级检索功能。

(4) 顾客在浏览图书时，可以方便地将图书放入自己的购物车中。

(5) 购物车中允许顾客随时调整及查看购买图书的实际信息。

(6) 顾客所有的购书信息在生成订单前将会一直被保存，生成订单结束后，自动清除相关信息。

(7) 系统采用会员制，会员采用唯一的顾客标识号来标识身份；会员可以修改个人信息，管理员可以对会员资料进行添加与删除。

(8) 管理员可以创建与维护图书类别。

(9) 管理员可以创建与维护图书概要信息及详情信息。

(10) 系统具有友好性和易操作性。

(11) 系统具有安全性和保密性。

1.1.5 需求分析人员分析用户的需求

第一步：细化并分析用户需求

对比较复杂的用户需求进行建模分析，以帮助软件开发人员更好地理解需求。

第二步：撰写产品需求规格说明书

需求分析人员按照指定的文档模板撰写《产品需求规格说明书》。如果待开发的产品分为软件和硬件两部分，则应当撰写《软件需求规格说明书》和《硬件需求规格说明书》。

第三步：进行需求确认

1. 需求建模

(1) 目标模型

步骤

第一步：建立业务目标到软件功能目标的转化模型

软件功能目标如图 1-2 所示。

第二步：建立业务限制因素到软件非功能目标的转化

软件非功能目标如图 1-3 所示。

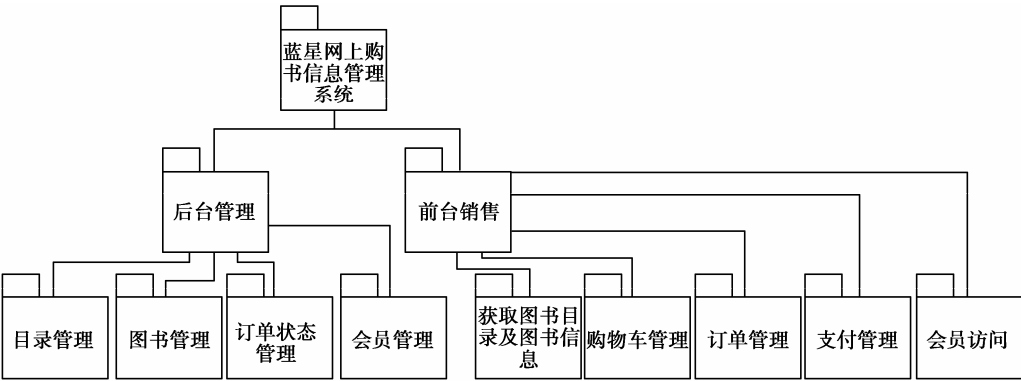


图 1-2 功能目标

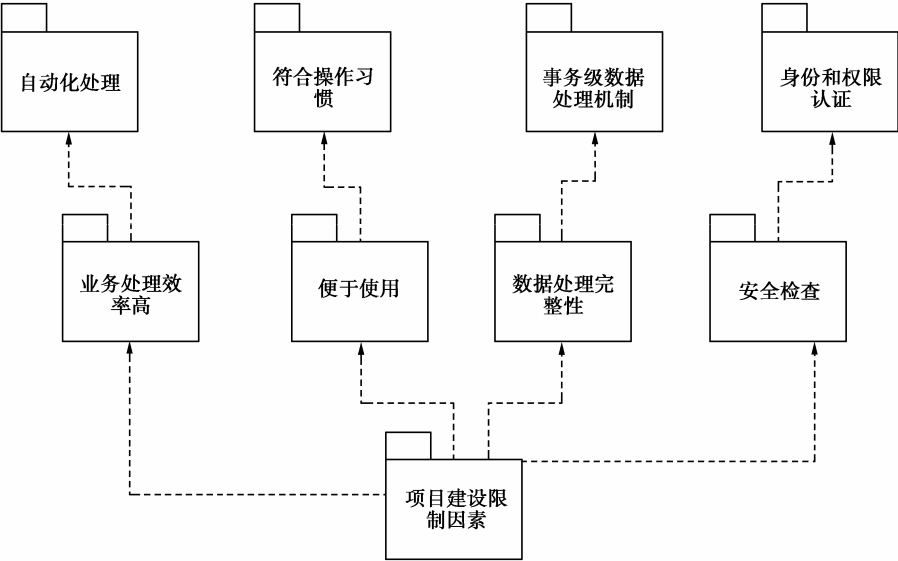


图 1-3 非功能目标

第三步：建立软件功能目标与非功能目标之间的双向束定关系
经过综合，得到以下关系模型，如图 1-4 所示。

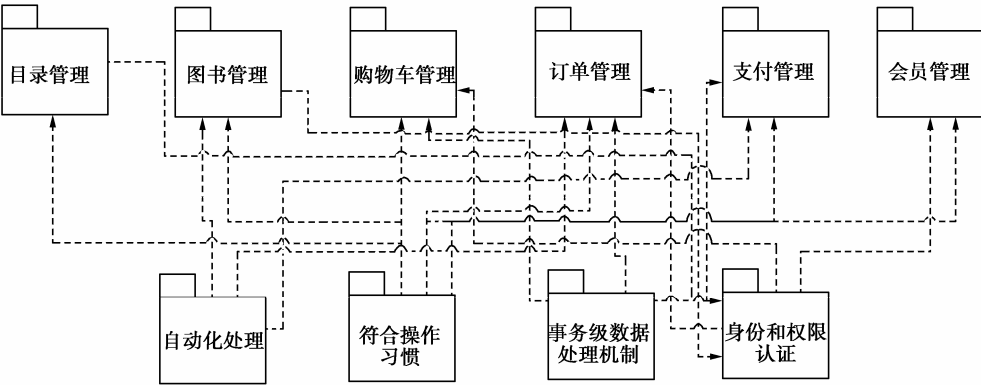


图 1-4 束定关系

(2) 用例模型

用例模型是系统既定功能及系统环境的模型，它可以作为客户和开发人员之间的契约。用例是贯穿整个系统开发的一条主线。一个用例模型包括了系统的所有用例，它是系统所有可能用途的总和。

蓝星网上购书信息管理系统用例一览表如表 1-1 所示。

表 1-1 蓝星网上购书信息管理系统用例一览表

目标 角色	FG1: 目录管理	FG2: 图书管理	FG3: 购物车管理	FG4: 订单管理	FG5: 支付管理	FG6: 会员管理
管理员	FG1: UC1: 目录创建 FG1: UC2: 目录编辑 FG1: UC3: 目录撤销 FG1: UC4: 目录查看	FG2: UC1: 图书入库 FG2: UC2: 图书出库 FG2: UC3: 图书撤销 FG2: UC4: 图书检索 FG2: UC5: 图书概要信息查看 FG2: UC6: 图书详细信息查看		FG4: UC3: 订单状态编辑 FG4: UC4: 订单信息查看 FG4: UC5: 订单检索 FG4: UC6: 订单撤销 FG4: UC7: 订单明细信息查看 FG4: UC8: 订单状态信息查看		FG6: UC1: 会员注册 FG6: UC2: 会员撤销
会员	FG1: UC4: 目录查看	FG2: UC5: 图书概要信息查看 FG2: UC6: 图书详细信息查看	FG3: UC1: 购物车图书添加 FG3: UC2: 购物车图书计价 FG3: UC3: 购物车图书数量更新 FG3: UC4: 购物车图书信息浏览 FG3: UC5: 购物车图书移除 FG3: UC6: 购物车图书清空	FG4: UC1: 订单生成 FG4: UC2: 个人订单信息查看	FG5: UC1: 信用卡支付	FG6: UC3: 个人资料修改 FG6: UC4: 会员登录
游客	FG1: UC4: 目录查看	FG2: UC5: 图书概要信息查看 FG2: UC6: 图书详细信息查看				FG6: UC1: 会员注册
银行账户处理系统					FG5: UC2: 转账	

(3) 业务对象模型

业务对象模型如图 1-5 所示。

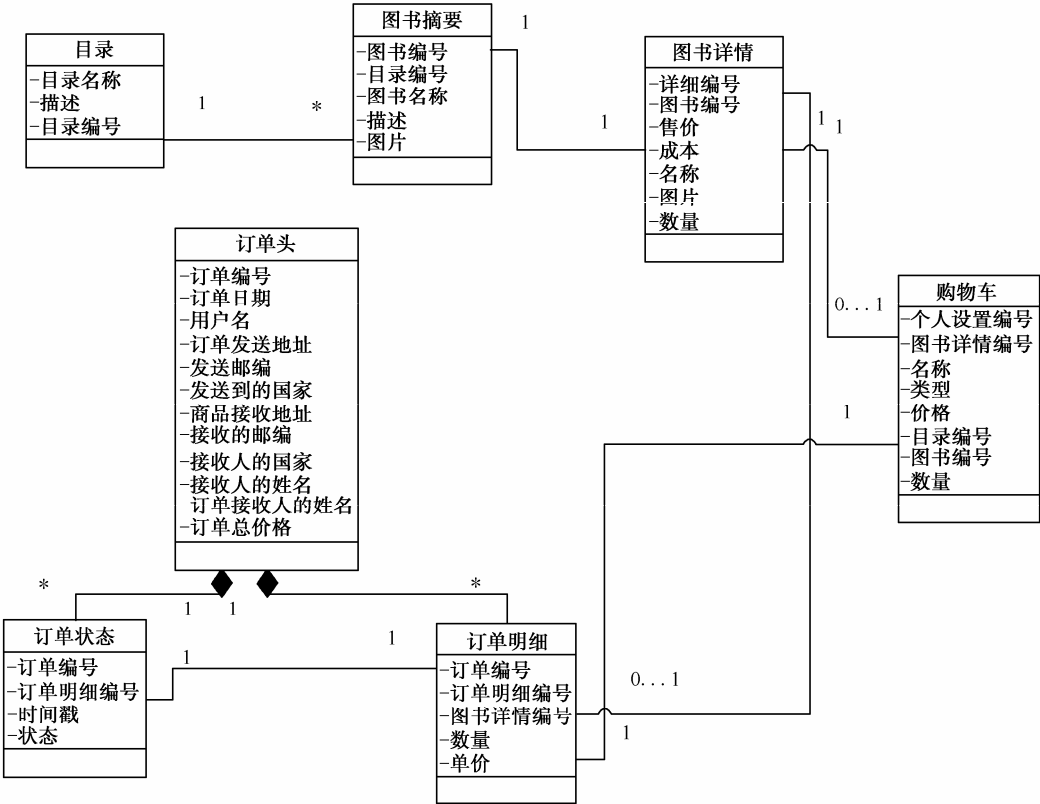


图 1-5 业务对象模型

2. 撰写需求规格说明书

《产品需求规格说明书》的重点是阐述“做什么”，而不是阐述“怎么做”。《产品需求规格说明书》应当正确、清楚、无二义性、一致、完备、可实现及可验证，“正确”是《产品需求规格说明书》最重要的属性。真正的困难是开发者和用户自己都不明白用户究竟“想要什么”和“不要什么”。为确保需求是正确的，开发方和用户必须对《产品需求规格说明书》进行确认；清楚的需求让人易读易懂，不在于文档的厚度。“无二义性”是指每个需求只有唯一的含义。如果一个人说的话，不同的人可能有不同的理解，那么这句话就有二义性。如果需求存在二义性，将会导致人们误解需求而开发出偏离需求的产品。为了使需求无二义性，人们在写《产品需求规格说明书》时措词应当准确，切勿模棱两可。“一致”（Consistent）是指《产品需求规格说明书》中各个需求之间不会发生矛盾。矛盾常常潜伏在需求文档的上下文中。“完备”（Complete）是指《产品需求规格说明书》中没有遗漏一些必要的需求。人们往往倾向于关注系统的特色功能，而忽视了一些不起眼的但却是必需的功能。“可实现”意味着在技术上是可行的，并且满足时间、费用、质量等约束。经过双方确认的《产品需求规格说明书》相当于商业合同，如果开发方不能够实现《产品需求规格说明书》中的内容，那就是违约，可能会被罚款。《产品需求规格说明书》中的各项需求对用户方而言应当都是可验证的（Verifiable）。如果需求是不可验证的，那么用户就无法验收软件，可能会发生商业纠纷。

产品需求规格说明书

0. 文档介绍

0.1 文档目的

对网上购书信息管理系统的业务、对系统要实现的主要功能性需求、非功能性需求等进行全面的阐述，以便帮助用户判断所要开发的软件是否符合其要求。该说明书将在软件开发目标和需求方面为用户和开发者之间创建一个共同的基础和共识，为后续的开发工作起到较好的指导作用。

0.2 文档范围

本需求规格说明书的主要内容有：介绍项目的背景，概述项目的任务与功能需求、性能需求及运行需求等内容。为业务上下文和系统功能建模，在完整的用例模型中记录系统需求。

0.3 读者对象

该文档可提供给用户代表、研发人员、需求测试人员、需求分析评审委员会专家、质保人员等相关人员阅读。

0.4 参考文档

- (1) 《计算机软件需求说明编制指南》——GB—9385—88
- (2) 《计算机软件产品开发文件编制指南》——GB—8567—88

0.5 术语、缩写及解释

术语、缩写	解 释
购物车（购物篮）	指的是应用于网上商城的在线购买功能，是为会员提供的一种快捷购物工具，它类似于大家在超市购物时使用的推车或篮子，用户可以免费把商品放入购物车，可以把购物车中的商品重新删除或更改购买数量等。通过购物车，用户可以在网上商城一次购买多件商品，并只需要进行一次结款
网上订单	通过网络进行填写、与商家签署的商业交易往来的文书

1. 产品介绍

(1) 产品名称：蓝星网上购书管理系统。

用途：在线购书，在线支付，产品展示，会员注册等。

(2) 产品的开发背景：网络经济是用现代信息技术和网络技术，依靠形成的互联网网络进行商务活动，它集金融电子化、管理信息化、办公室自动化于一体，与传统商务模式相比较，具有无可比拟的运作优势。

- ① 便于广告宣传。
- ② 运营成本低。
- ③ 用户范围广。
- ④ 商务开放性好。
- ⑤ 无时空限制。
- ⑥ 多媒体手段表现力强。
- ⑦ 以顾客为中心，最大限度地满足顾客的个性化需求。
- ⑧ 提升企业形象。

- ⑨ 促进市场营销。
- ⑩ 增强企业管理功能。
- ⑪ 革新企业管理思想。

足不出户就能买到心仪的图书，是越来越多上网爱好者实现购物的一种手段，为了满足长春职业技术学院学生的购书需要，长春职业技术学院推出了自己的电子商务系统——蓝星网上购书管理系统。

本项目分为前、后台两个管理系统。

前台销售管理系统实现的功能类似于现实生活中的商店销售，顾客可以浏览图书、选择图书、结算完成购书等。图书数据能根据需要灵活地检索与显示，简化购物流程，真正做到“简捷、高效、流畅”的购物环境。

本系统的后台管理系统融入企业传统的进销存概念，对订单进行管理，对图书进行分类管理，提供图书信息的添加、修改和删除等功能，支持图书的封面图片上载功能，从而能够利用网络的优势增强图书商品的宣传效果。

2. 产品面向的用户群体

顾客群主要由长春职业学院在校的近万名学生，以及 Internet 上来自全国各地的访问该网上书店的潜在客户组成。

3. 产品应当遵循的标准或规范

- (1)《电子商务模式规范》——商务部，2008 年 5 月
- (2)《网络购物服务规范》——商务部，2008 年 5 月

4. 产品范围

本产品适用于中小型图书电子商务垂直网站。

5. 产品中的角色

角 色 名 称	职 责 描 述
管 理 员	书店员工，对网上书店网站内容进行后台更新
会 员	在书店登记注册的顾客，通过唯一的身份标识可以在线购书
游 客	通过 Internet 访问网上书店的人，可以浏览图书目录信息及图书信息

6. 产品的功能性需求

6.1 功能性需求分类

功 能 类 别	功能名称、标识符	描 述
FG1：目录管理	FG1：UC1：目录创建	管理员添加图书类别
	FG1：UC2：目录编辑	管理员修改图书类别
	FG1：UC3：目录撤销	管理员删除图书类别
	FG1：UC4：目录查看	获取、查看图书类别信息

功 能 类 别	功能名称、标识符	描 述
FG2：图书管理	FG2： UC1： 图书入库	管理员添加图书概要信息、详细信息及库存数量
	FG2： UC2： 图书出库	图书售出后更新图书库存数量
	FG2： UC3： 图书撤销	管理员删除图书所有信息
	FG2： UC4： 图书检索	依据关键字段及值，获得图书概要信息
	FG2： UC5： 图书概要信息查看	获得、查看图书概要信息
	FG2： UC6： 图书详情查看	查看图书详细信息
FG3：购物车管理	FG3： UC1： 购物车图书添加	具有会员资格的顾客添加要购买的图书到购物车中
	FG3： UC2： 购物车图书计价	计算购物车中的图书总价格
	FG3： UC3： 购物车图书数量更新	顾客更新购物车中某图书的数量
	FG3： UC4： 购物车图书查看	具有会员资格的顾客浏览购物车内所有的图书信息
	FG3： UC5： 购物车图书移除	具有会员资格的顾客删除购物车里的图书
	FG3： UC6： 购物车图书清空	清空购物车
FG4：订单管理	FG4： UC1： 订单生成	具有会员资格的顾客下订单
	FG4： UC2： 个人订单信息查看	具有会员资格的顾客查看自己所下的所有订单信息
	FG4： UC3： 订单状态编辑	管理员依据交货进度，修改订单状态
	FG4： UC4： 订单查看	管理员查看所有订单信息
	FG4： UC5： 订单检索	管理员依据关键字段及值，查看订单信息
	FG4： UC6： 订单撤销	管理员删除订单
	FG4： UC7： 账户地址获得	获得会员保存的订单发送地址
	FG4： UC8： 账户地址保存	会员保存自己的订单接收地址，方便下次购买商品时使用
	FG4： UC9： 订单明细查看	获得订单明细信息
	FG4： UC10： 订单状态查看	获得订单状态信息
FG5：结算管理	FG5： UC1： 信用卡信息录入	会员录入信用卡信息支付购买款项
	FG5： UC2： 银行转账	调用银行转账系统完成
FG6：会员管理	FG4： UC1： 会员资格注册	管理员添加会员信息或游客申请注册会员
	FG4： UC2： 会员资格撤销	管理员删除会员
	FG4： UC3： 会员资料编辑	会员可对个人信息进行修改

6.2 功能模块说明
(略)

7. 产品的非功能性需求

7.1 用户界面需求

需 求 名 称	详 细 要 求
用户界面标准	符合微软 GUI 标准
用户的交互方法	对常用功能定义快捷键及右键菜单
屏幕行为	进入程序后，焦点默认于左上角的第一个控件，可使用 Tab 键切换焦点

7.2 软硬件环境需求

1. 服务器端

需 求	详 细 要 求
操作系统	MS Windows Server 2003/2008 标准版/企业版
脚本解释器	Javascript1.5 版本，安装 IE5.5 以上版本即可获得
Web 服务器	IIS5.0 以上
数据库服务器	MS SQL Server 2000/2005 标准版/企业版
权限要求	对 SQL Server 数据库具有建表、备份的权限
硬件要求	服务器 CPU*2 4GB 内存 146G*3 SAS 硬盘 Raid5
空间大小	初次安装至少需 10MB 可用空间

2. 客户端

需 求	详 细 要 求
浏览器	IE5.0 以上
分辨率	最佳效果 1 024×768

7.3 产品质量需求

主要质量属性	详 细 要 求
正确性	软件能够正确执行任务，工作成果准确
健壮性	能够对异常情况作出处理，不死机，并能够保持数据的完整性
可靠性	平均故障间隔时间>3 个月，99.9%系统故障在 1 小时内可修复
性能，效率	吞吐量≥25 笔事务/秒，容量：10 万客户，响应时间<5s
易用性	用户界面风格一致，提供在线演示，员工经过简单培训即可上手，符合 GUI 标准
清晰性	程序及文档注释详尽，说明文字简洁易读，易理解
安全性	身份认证，权限认证，关键数据加密处理
可扩展性	基础架构可横向与纵向扩展
兼容性	支持多语言开发的组件

7.4 其他需求

（略）

8. 需求确认

项目经理邀请同行专家和用户（包括客户和最终用户）一起评审《产品需求规格说明书》，尽最大努力使《产品需求规格说明书》能够正确无误地反映用户的真实意愿。

需求评审之后，开发方和客户方的责任人对《产品需求规格说明书》作书面承诺。

1.1.6 任务小结

开发软件系统最困难的部分就是准确说明开发什么，最困难的概念性工作是编写出详细的需求，包括所有面向用户、面向机器和其他软件系统的接口。此工作一旦做错，将会给系统带来极大的损害，并且以后对其修改也极为困难。

需求是产品的根源，需求工作的优劣对产品的影响最大。就像一条河流，如果源头被污染了，那么整条河流也就被污染了。

需求开发的目的是通过调查与分析，获取用户需求并定义产品需求。

需求调查的目的是通过各种途径获取用户的需求信息（原始材料），产生《用户需求说明书》。

需求分析的目的是对各种需求信息进行分析，消除错误，刻画细节等。常见的需求分析方法有“问答分析法”和“建模分析法”两类。

需求定义的目的是根据需求调查和需求分析的结果，进一步定义准确无误的产品需求，产生《产品需求规格说明书》。系统设计人员将依据《产品需求规格说明书》开展系统设计工作。

需求管理的目的是在客户与开发方之间建立对需求的共同理解，维护需求与其他工作成果的一致性，并控制需求的变更。

需求确认是指开发方和客户共同对需求文档进行评审，双方对需求达成共识后做出书面承诺，使需求文档具有商业合同的效果。

需求跟踪是指通过比较需求文档与后续工作成果之间的对应关系，建立与维护“需求跟踪矩阵”，确保产品依据需求文档进行开发。

需求变更控制是指依据“变更申请—审批—更改—重新确认”的流程处理需求的变更，防止需求变更失去控制而导致项目发生混乱。

在完成蓝星网上购书信息管理系统用户需求的分析与处理的任务中，我们遵循“初始、细化、构造与移交四步走”的路线，主要完成用户需求的收集、需求建模及《产品需求规格说明书》的撰写等。

在进行需求开发的过程中会面临很多困难，许多开发人员更是错误地以为：需求是用户的事情，不是我们的事情。我们为用户开发软件，难道用户不该告诉我们应当开发什么吗？如果用户说不清楚需求，或者经常变更需求，这类问题是用户产生的，应当由他们自己负责。

用户说不清楚需求或者需求发生变更，这些都是常见的问题，并非不可克服，是人们可以设法解决的。可悲的是开发人员把这些问题当成了借口，不愿主动攻克问题，导致需求问题扩散到整个软件开发过程中，产生太多的后患。

需求分析人员的天职就是在有限的时间内获取准确而细致的用户需求，如果做不到就是失职，不能找借口。

1.1.7 练习题

- 1. 如何进行需求分析？
- 2. 什么是好的需求规格说明书？

1.2 项目计划安排

学习目标

- 理解软件项目计划的作用与主要组成要素
- 初步掌握如何制定中小型软件项目计划
- 建议学时：4 学时

1.2.1 任务名称：项目计划安排

本任务名称为蓝星网上购书管理信息系统项目计划安排。

1.2.2 任务描述

编写蓝星网上购书信息管理系统用于协调所有项目计划编制的文件、指导项目执行和控制的文件。要清楚地描述出：

- 项目划分的各个实施阶段；
- 每个阶段的工作重点和任务是什么；
- 完成本阶段工作和任务的人力、资源需求，时间期限；
- 阶段工作和任务的成果形式；
- 项目实施过程中对风险、疑难、其他不可预见因素等的处理机制；
- 各任务组及开发人员之间的组织、协调关系等。

1.2.3 任务分析

根据《GB8567—88 计算机软件产品开发文件编制指南》中项目开发计划的要求，结合实际情况调整后的《项目计划书》主要内容索引如下：

- 项目概述
- 项目目标
- 产品目标与范围
- 假设与约束
- 项目工作范围
- 应交付成果
- 需完成的软件
- 需提交用户的文档
- 须提交内部的文档
- 应当提供的服务
- 项目开发环境
- 项目验收方式与依据

项目团队组织

- 组织结构
- 人员分工
- 协作与沟通
- 内部协作
- 外部沟通

实施计划

支持条件

预算（可选）

关键问题

专题计划要点

以上述提纲为泛本，完成蓝星网上购书信息管理系统的编写。

1.2.4 项目概述

本产品应是一个具有灵活性和系统性，可扩充性和可维护性强，可以不断延续发展的电子商务软件，其包括的内容有：

- 在对网上购书信息管理系统总体业务进行分析的基础上进行提炼，充分考虑系统性与可扩充性；
- 遵循各种国际、国家及行业标准；
- 综合考虑与外部系统（银行等）的接口；
- 系统必须实用、友好、稳定、可靠、可移植性好、可扩充性好；
- 方便实施与维护，能减轻客户化工作；
- 建立新型的业务与技术模型，注重业务流程的重用性与可定制性。用户能根据自身的需要方便地定义，采用基于三层框架的内核，并以内核为框架，采用面向组件对象的方式，建立以面向组件为基础的结构化的综合应用体系。灵活地实现应用对象的重组，降低维护管理的成本；同时基于接口技术的应用可以使得体系更加灵活和便于扩充，具备“平台”的概念。

1.2.5 主要参加人员

参加该项目计划安排的主要人员如图 1-6 所示。

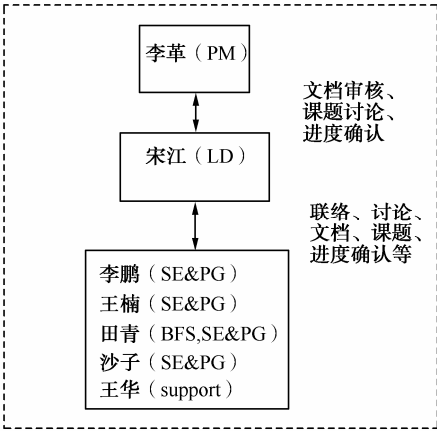


图 1-6 主要参加人员

1.2.6 应交付成果

1. 程序（见表 1-2）

表 1-2 交付程序

存储程序的媒体	光盘
程序功能概述	主要实现图书信息在线浏览、放入购物车、在线下订单及完成支付的功能

2. 文件（见表 1-3）

表 1-3 交付文件

系 统 名 称	文 件
蓝星网上购书信息管理系统	需求规格说明书 用户手册

3. 服务（见表 1-4）

表 1-4 提供服务

系 统 名 称	服 务
蓝星网上购书信息管理系统	两年免费售后服务（开发新功能费用另行计算）

4. 非移交的产品（见表 1-5）

表 1-5 非移交产品

系 统 名 称	产 品
蓝星网上购书信息管理系统	概要设计说明书 数据库设计说明书 详细设计说明书 模块开发说明 测试报告

1.2.7 验收标准

验收标准见表 1-6。

表 1-6 验收标准

系 统 名 称	验 收 方 式	验 收 依 据
蓝星网上购书信息管理系统	交付后验收	需求规格说明书

1.2.8 完成项目的最迟期限

该项目完成的最迟期限（即交付期限）见表 1-7。

表 1-7 交付期限

系 统 名 称	期 限
蓝星网上购书信息管理系统	2008 年 8 月 25 日

1.2.9 实施计划

1. 工作任务的分解与人员分工（见表 1-8）

表 1-8 工作任务分解与人员分工

系 统 名 称	工 作	所 需 人 员	所 需 天 数	标志性事件（交付物）
蓝星网上购书信息管 理系统	需求分析	2	3	需求规格说明书
	系统设计	2	3	概要设计说明书

续表

系 统 名 称	工 作	所 需 人 员	所 需 天 数	标志性事件（交付物）
蓝星网上购书信息管理系统	子系统设计	4	5	详细设计说明书
	系统编码与实现	4	15	可测试代码
	系统测试	4	3	测试报告
	系统实施与维护、使用培训	4	3	用户手册 验收报告

2. 前期项目计划表

使用 Project 软件制作一个前期的项目计划表及甘特图，如图 1-7 所示。随着开发工作的深入，该项目计划表会不断地被加以细化和补充。

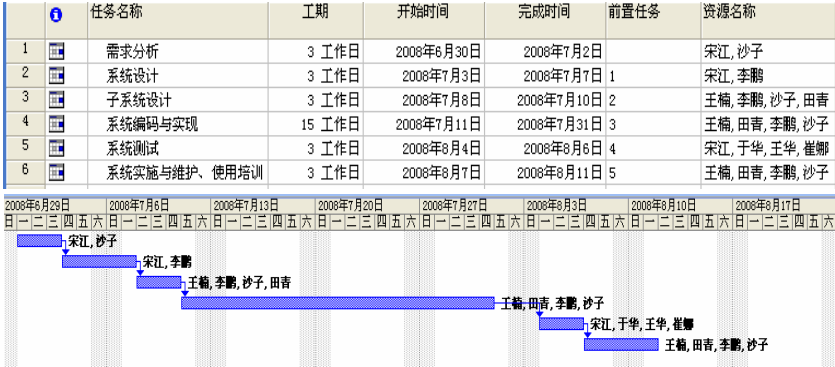


图 1-7 项目计划表及甘特图

3. 预算（见表 1-9）

表 1-9 预算

项 目	工 作 量	价 格	金额（万元）
蓝星网上购书信息管理系统	4.0 人/月	每人月 4 000 元	1.6

1.2.10 系统运行软硬件环境

1. 硬件环境

在初期访问压力不大的情况下，Web 服务器和数据库服务器使用一台机器即可，配置如表 1-10 所示。

表 1-10 硬件环境

配 件	配 置
CPU	至强服务器 CPU*2
内存	4GB
硬盘	146GB*3 SAS 硬盘 Raid5

如果后期访问压力增大，建议安装一台同样配置的服务器，将 Web 服务器和数据库服务器部署到不同的机器上。

2. 软件环境（见表 1-11）

表 1-11 软件环境

操 作 系 统	MS Windows Server2003/2008 标准版/企业版
数 据 库	MS SQL Server2000/2005 标准版/企业版

3. 网络环境（见表 1-12）

表 1-12 网络环境

环 境	备 注
100MB 网络接入	建议接在校园主干网上，若需对校外发布，则需要公网 IP
防火墙	建议采用硬件防火墙，可以考虑使用现有资源

1.2.11 任务小结

软件项目计划是一个软件项目进入系统实施的启动阶段，通过该工作环节可以确定详细的项目实施范围，定义递交的工作成果，评估实施过程中主要的风险，制定项目实施的时间计划、成本和预算计划、人力资源计划等。

在这其中，进度安排更是软件项目计划的首要任务，而项目计划则是软件项目管理的首要组成部分。与估算方法和风险分析相结合，进度安排将为项目管理者建立起一张计划图。

常用的制定进度计划的工具主要有 Gantt 图和工程网络两种。Gantt 图具有历史悠久、直观简明、容易学习、容易绘制等优点，但是，它不能明显地表示各项任务彼此间的依赖关系，也不能明显地表示关键路径和关键任务，进度计划中的关键部分不明确。因此，在管理大型软件项目时，仅用 Gantt 图是不够的，不仅难以做出既节省资源又保证进度的计划，而且还容易发生差错。

1.2.12 练习题

如何编制软件项目进度计划表？

1.3 系统设计

学习目标

- 理解系统架构的主要作用
- 初步掌握运用 RUP4+1 视图指导完成系统架构的设计
- 建议学时：4 学时

软件架构是指一个系统的基础组织，具体体现在系统的组成构件，构件之间、构件和环境之间的关系，以及指导其设计和演化的原则上。

1.3.1 任务名称：系统设计

1.3.2 任务描述

设计蓝星网上购书信息管理系统架构。

1.3.3 任务分析

RUP 的整个分析与设计过程如图 1-8 所示，其中架构分析人员即为软件架构师角色，从下图中可见系统架构从分析到设计的全部活动。

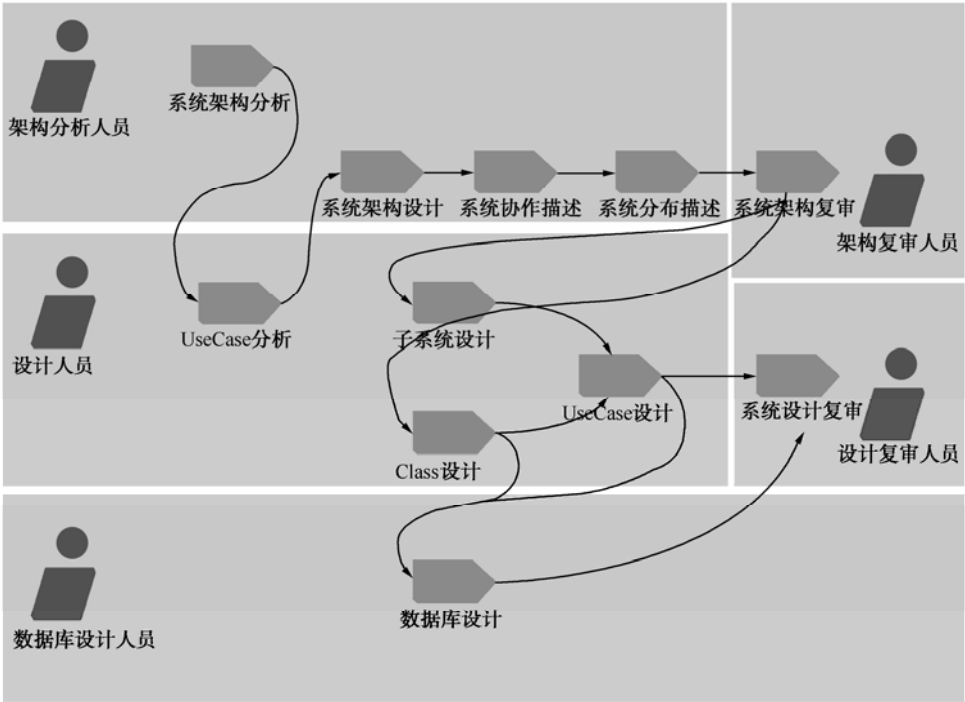


图 1-8 系统架构分析与设计活动

系统架构要涵盖的内容和决策很多，超过了人脑“一蹴而就”的能力范围，因此采用“分而治之”的办法从不同视角分别设计；同时，也为软件架构的理解、交流和归档提供方便，所以引入软件架构视图来完成系统架构的设计。

一个架构视图是对于从某一视角或某一点上看到的系统所做的简化描述，描述中涵盖了系统的某一特定方面，而省略了与此方面无关的实体。

视图方法是软件架构归档的方法，更是指导我们进行架构设计的思维方法。

开发全景中的构架视图如图 1-9 所示。

1995 年，Philippe Kruchten 提出的 4+1 视图方法如图 1-10 所示。

该方法的不同架构视图承载不同的架构设计决策，支持不同的目标和用途。

逻辑视图：当采用面向对象的设计方法时，逻辑视图即为对象模型。

开发视图：描述软件在开发环境下的静态组织。

处理视图：描述系统的开发和同步方面的设计。

物理视图：描述软件如何映射到硬件，反映系统在分布方面的设计。

运用 4+1 视图方法，可针对不同需求进行架构设计。

要开发出用户满意的软件并不是件容易的事，软件架构师必须全面把握各种各样的需求、权衡需求之间有可能的矛盾之处，分门别类地将不同需求一一满足。

ViewPoint 视角 Granularity 粒度	UseCase 用例规格	Logical & Process			Implementation 实施（物理）	Deployment 部署
		Analysis Logic 分析逻辑	Design Logic 设计逻辑	Runtime 运行时刻		
Architectural Level	UseCase Packages/ UseCases Structure/	Service Packages/ Key Abstraction Classes	Layers/ Packages/ Subsystems/ Classes	Processes/ Threads Structure	Component Packages/ Components	Topological Structure/ Node Configurations
	Key UseCases Spec	Analysis Mechanlsm	Design Mechanism		Implementation Mechanism	
Subsystem Level		Analysls Classes	Classes/ Subsystems	Containing of Classes/ Subsystems	Interfaces of Component/ Assignment Of Clesses	
	UseCase Spec.	UseCase Analysis	UseCase Design			
Class Level		Responsibilities	Features/Methods			

图 1-9 开发全景中的构架视图

Philippe Kruchten 提出的 4+1 视图方法为软件架构师“一一征服需求”提供了良好的基础，如图 1-11 所示。

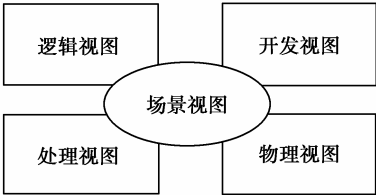


图 1-10 4+1 视图

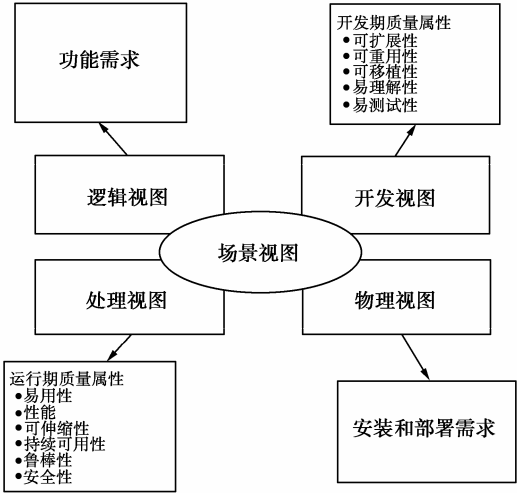


图 1-11 4+1 视图含义

逻辑视图：逻辑视图关注功能，不仅包括用户可见的功能，还包括为实现用户功能而必须提供的“辅助功能模块”；它们可能是逻辑层、功能模块等。

开发视图：开发视图关注程序包，不仅包括要编写的源程序，还包括可以直接使用的第三方 SDK 和现成框架、类库，以及开发的系统将运行于其上的系统软件或中间件。开发视图和逻辑视图之间可能存在一定的映射关系，如逻辑层一般会映射到多个程序包等。

处理视图：处理视图关注进程、线程、对象等运行时的概念，以及相关的并发、同步、

通信等问题。处理视图和开发视图的关系为：开发视图一般偏重程序包在编译时期的静态依赖关系，而这些程序运行起来之后会表现为对象、线程、进程，处理视图比较关注的正是这些运行时单元的交互问题。

物理视图：物理视图关注“目标程序及其依赖的运行库和系统软件”最终如何安装或部署到物理机器，以及如何部署机器和网络来配合软件系统的可靠性、可伸缩性等要求。物理视图和处理视图的关系为：处理视图特别关注目标程序的动态执行情况，而物理视图重视目标程序的静态位置问题；物理视图是综合考虑软件系统和整个 IT 系统相互影响的架构视图。

如前所述，下面运用 RUP 4+1 视图完成蓝星网上购书信息管理系统架构的设计。

1.3.4 层图（逻辑视图）

层图如图 1-12 所示。

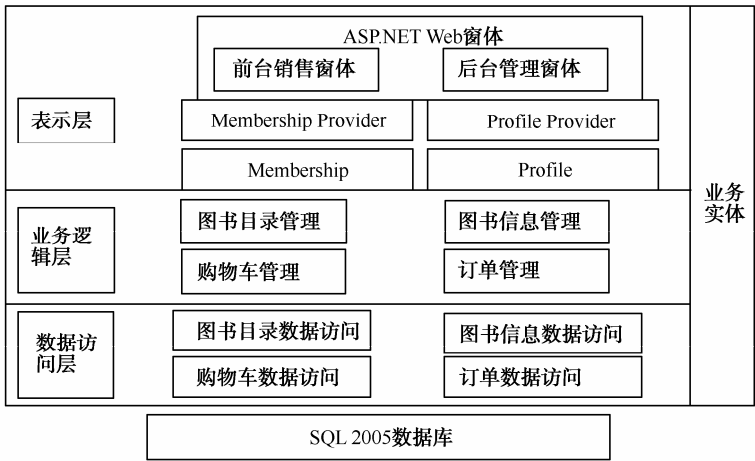


图 1-12 层图

1.3.5 包图（开发视图）

包图如图 1-13 所示。

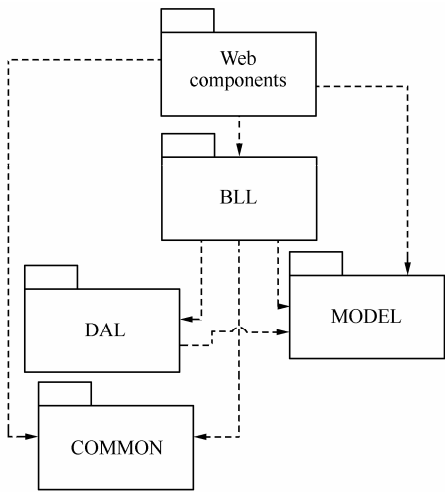


图 1-13 包图

1.3.6 部署图（物理视图）

部署图如图 1-14 所示。

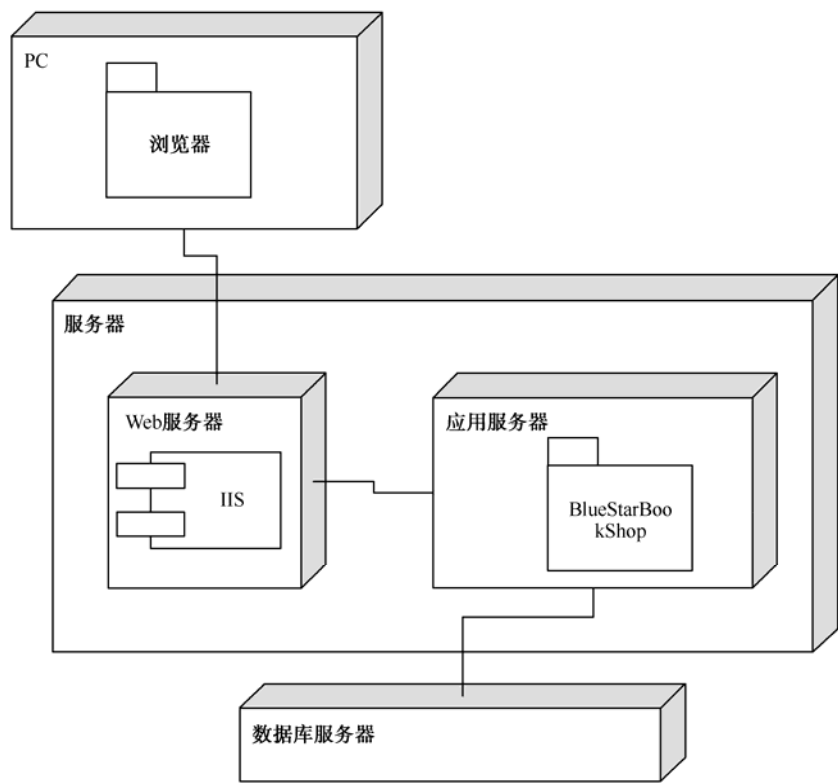


图 1-14 部署图

1.3.7 选择技术

- 开发环境：Microsoft Visual Studio 2008 集成开发环境
- 编程语言：ASP.NET + C#
- 中间件：Microsoft DotnetFrameWork 3.5
- 数据库：Microsoft SQL Server 2005

1.3.8 安全策略

网上书店的实施，其关键是要保证整个商务过程中系统的安全性。
实现网上书店的关键是要保证商务活动过程中系统的安全性，即保证基于互联网的电子交易过程与传统交易的方式一样安全可靠。

1. 密码技术

采用 MD5 加密。

2. 访问控制

采用授权策略和机制。保护可以从以下几个方面加以考虑：物理隔离、时间隔离、密码隔离。

3. 防火墙技术

采用分组过滤防火墙技术。

1.3.9 并发策略

① 对于业务数据低级并发控制由数据库事务和线程监视器自动管理，对业务服务的每一次使用都封装在一个事务中，以正确传送给数据库管理系统。

② 会员不可以重复登录，只能登录一次。

③ 在脱线状态下更新数据库数据，在凌晨使更新真正生效。

1.3.10 任务小结

为什么需要软件架构，这是因为最终开发出来的目标系统总是由多具组成部分所构成的，这种结构如果没有预先定义，很难保证系统的构建过程能自发创建出一个一致而满足需求的交付。当前的软件规模已大到需要采用团队开发的模式，多个开发人员的分工协作，必然依赖于一种对开发内容的合理划分，以减少相互干扰、缩短工期的关键路径，从而提高开发效率、加快项目进度——软件架构无疑是其中最关键的一类划分，它将被用来计划、管理与执行系统开发的各项活动。目标系统总是要面临各种变数，项目组期望系统在发生变更、部署到新环境中时，仍然保持既有的稳定、可靠的性能——目标系统应具备一种健壮性。系统的构建要经历一个不断增添新功能、加入新行为的过程，项目组期望做得比较容易、开销较低，且在此过程中不存在重大的风险——目标系统应具备一种可扩展性。这些质量属性归根结底要落实到软件架构之上。

软件架构是增量式开发的前提，迭代生命周期模型开始逐渐代替传统的瀑布模型，然而要真正实现增量式开发的目标，却需要满足若干关键的前提条件：向已有交付添加新功能非常容易（可扩展）；后续的增量不会破坏已有的交付，使得迭代退化为返工（健壮），这些条件最终归结为在大批量的增量开发之前，要构建一个架构基线，它同时提供可扩展性与健壮性。

设计良好的对象可以方便地添加新的行为，而封装性为其应对变化提供免疫力，基于对象的架构在微观上便具有更强的可扩展性与健壮性。分层（分包、子系统）架构在大粒度上隔离关注面，同样从宏观上增强了可扩展性与健壮性。

软件架构也是实现健壮性与可扩展性的途径。要实现健壮与可扩展性等质量特性，主要有两个途径——尽可能降低系统的冗余程度，同时隔离不同的关注面（实质是高内聚、低耦合，例如：将稳定部分与可变部分隔离，将用户交互与业务、数据等功能域分离，将功能和非功能的实施代码分离）。隔离关注面，使得扩展或变更时，对系统的修改局部化，对其他部分造成的影响被限制在较小范围内，避免出现那种牵一发而动全身的情形；高内聚的结构也利于聚焦于各部分的设计适应性上。低冗余，使得即使要变更，变更所触及的部分也尽可能地少；系统被改动的地方越少当然就越健壮，同时开销也小、实施也更容易。

1.3.11 练习题

1. 为什么要开发软件架构？
2. 架构设计的主要内容是什么？

1.4 子系统设计

学习目标

- 理解如何设计类与实现业务服务
- 掌握如何将运行对象映射为可存储对象
- 建议学时：12 学时

1.4.1 任务名称：子系统设计

1.4.2 任务描述

完成蓝星网上购书信息管理系统子系统设计，包括类的设计、用例具体实现、用户界面设计及数据库设计。

1.4.3 任务分析

通过子系统设计形成一个可用的、完整的解决方案，并且能够比较容易地将方案转换成程序代码。该任务在 .NET 标准三层系统架构的基础上，将考虑所有的实现技术问题，对分析阶段的模型进行扩展和细化，并对分析阶段定义的类进一步扩充，定义新的类来处理技术方面的问题，最终形成最后的解决方案。

1. 遵循类的设计原则

开闭原则：对扩展开放而对变更封闭。

依赖倒置原则：依赖抽象类而非具体类。

Liskov 替换原则：子类应当能完全替代其基类。

单一职责原则：一个类只应当承担单一和集中的职责，这样引发类进行变更的原因只有一个。

接口分离原则：为客户提供多个特定的接口好过一个多种用途集于一身的接口，即客户不被强制依赖于其不需要的操作。

组合复用原则：尽可能地使用对象的多态组合而非继承来实现复用。

所知最少原则：一个类的操作实现中，只应调用下列对象的操作——它自己、作为参数传入的对象、它创建的对象、它包含的对象。

2. 实现系统用例

实现系统用例的方式通常显示出用例如何通过一系列协作类进行实现，这是系统内部行为的模型，它可以用两个 UML 工件来描述：实现类图和顺序图。

3. 用户界面设计

在每个用户界面中，都应提供对应各个用例的窗口，在用户界面和用例之间应具有清晰的映射关系。

4. 数据库设计

面向对象的数据库设计是从对象模型出发的，属于实体主导型设计，数据库设计（模式）是否支持应用系统的对象模型，这是判断是否是面向对象数据库系统的基本出发点。由于应用系统设计在前，数据库设计随后，所以应用系统对象模型向数据库模式的映射是面向对象数据库设计的关键。由于 RDBMS 是以二维表为基本管理单元的，所以对象模型最终是由二

维表及表间关系来描述的。换言之，对象模型向数据库概念模型的映射就是向数据库表的变换过程。有关的变换规则简单归纳如下。

一个对象类可以映射为一个以上的库表，当类间有一对多的关系时，一个表也可以对应多个类。

关系（一对一、一对多、多对多及三项关系）的映射可能有多种情况，但一般映射为一个表，也可以在对象类表间定义相应的外键。对于条件关系的映射，一个表至少应有 3 个属性。

单一继承的泛化关系可以对超类、子类分别映射表，也可以不定义父类表而让子类表拥有父类属性；反之，也可以不定义子类表而让父类表拥有全部子类属性。

对多重继承的超类和子类分别映射表，对多次多重继承的泛化关系也映射一个表。

对映射后的库表进行冗余控制调整，使其达到合理的关系范式。

1.4.4 类的列表

类的列表如表 1-13 所示。

表 1-13 类的列表

包	类 名	说 明
Web	Default	前台首页页面类
	BookBrief	前台图书概要页面类
	Item	前台图书详情页面类
	ShoppingCart	前台购物车页面类
	CheckOut	前台订单页面类
	UserProfile	前台个人设置页面类
	Search	前台图书检索页面类
	Register	前台会员注册页面类
Admin	Default	后台首页页面类
	Category	后台目录管理页面类
	Book	后台图书管理页面类
	Order	订单管理页面类
	Login	后台登录页面类
BLL	OrderManager	订单管理逻辑类
	CategoryManager	目录管理逻辑类
	BookBriefManager	图书概要管理逻辑类
	ItemManager	图书详情管理逻辑类
	CartManager	购物车管理逻辑类
	AccountManager	账户设置管理逻辑类
DAL	OrderAccess	订单数据访问类
	CategoryAccess	目录数据访问类
	BookBriefAccess	图书概要数据访问类
	ItemAccess	图书详情数据访问类
	CartAccess	购物车数据访问类
Model	OrderInfo	订单信息类
	CategoryInfo	目录信息类
	BookBriefInfo	图书概要信息类

包	类 名	说 明
Model	ItemInfo	图书详情信息类
	CartInfo	购物车信息类
	LineItemInfo	订单明细信息类
	OrderStateInfo	订单状态信息类
	AddressInfo	账户地址信息类
Common	DataBase	数据库连接类
	SQLString	SQL 语句构造类

1.4.5 类的规格说明示例

1. CartManager

CartManager 的字段见表 1-14。

表 1-14 CartManager 私有数据成员

字 段 名 称	类 型	说 明
CartItems	Dictionary<string, CartInfo>	private, 购物车图书集合
DAL	CartAccess	private static readonly, CartAccess 对象

CartManager 的属性见表 1-15。

表 1-15 CartManager 公共属性

属 性 名 称	类 型	说 明
Total	decimal	public, 购物车图书总价格
Count	int	public, 购物车中所选定图书项数
CartItems	ICollection<CartInfo>	public, 购物车中选购信息集合

CartManager 的方法见表 1-16。

表 1-16 CartManager 类公有方法

方 法 名 称	返回值类型	返回值说明	参 数	参 数 类 型	参 数 说 明	概 要
SetQuantity	void	空	bookId qty	string int	图书编号 购买数量	公有方法，设置所要购买图书的数量
Add	void	空	bookId	string	图书编号	公有方法，将选定图书放入购物车
Add	void	空	cartItem	cartInfo	购物车明细对象	公有方法，将 Cart 表中的一条记录数据添加到购物车中
Remove	void	空	bookId	string	图书编号	公有方法，从购物车中按书号移除图书
Clear	void	空				公有方法，清空购物车中的所有图书
GetCartItems	void	空	username	string	会员的用户名	公有方法，调用 CartAccess 类的同名方法
SetCartItems	void	空	username cartItems	string ICollection <CartInfo>	会员的用户名 购物车图书集合	公有方法，调用 CartAccess 类的同名方法

方法名称	返回值类型	返回值说明	参数	参数类型	参数说明	概要
GetOrderLineItems	LineItemInfo[]	订单明细对象数组				公有方法，将购物车中的图书信息转换成订单明细信息

2. CartAccess

CartAccess 类的公有方法见表 1-17。

表 1-17 CartAccess 类公有方法

方法名称	返回值类型	返回值说明	参数	参数类型	参数说明	概要
GetCartItems	IList<CartInfo>	Cart 表中指定会员的购物车信息集合	username	string	会员的用户名	公有方法，从 Cart 表中获得该会员所保存的购物车数据
SetCartItems	void	空	username cartItems	string Icollection <CartInfo>	会员的用户名 购物车图书集合	公有方法，用购物车中的图书信息更新 Cart 表中的数据，使二者保持一致

因篇幅有限，其他类的规格说明详情请参见相关书籍。

1.4.6 用例具体实现示例

1. 购物车图书添加（放入图书到购物车，见图 1-15 至图 1-17）

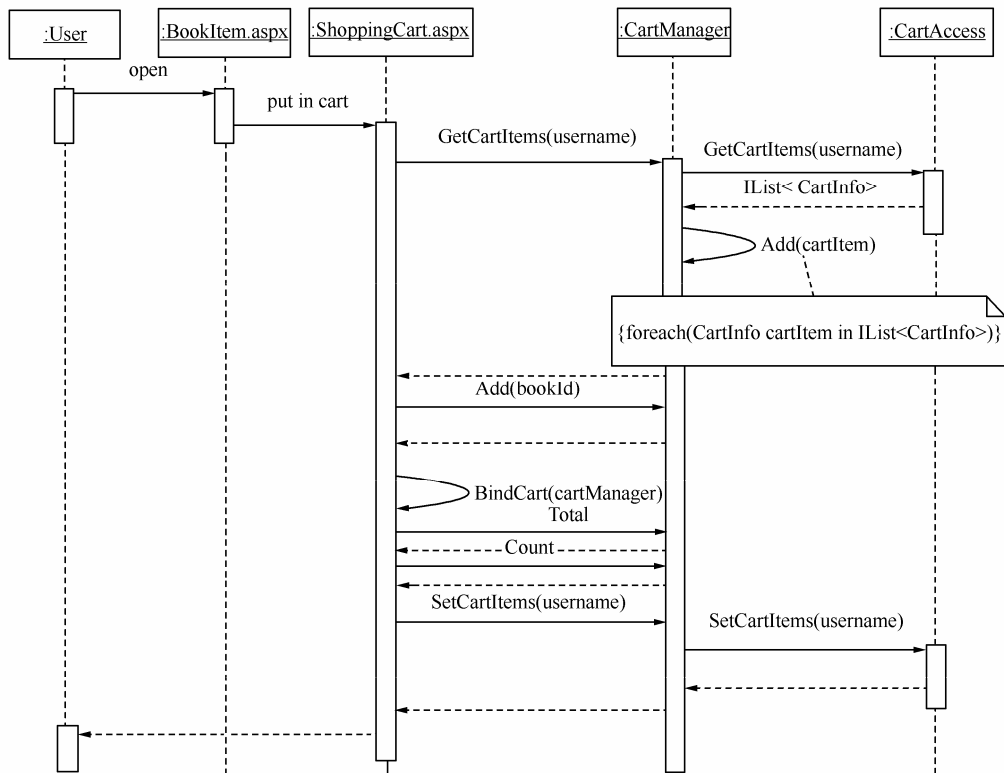


图 1-15 购物车图书添加序列图



图 1-16 图书详情页面



图 1-17 购物车页面

2. 订单生成 (见图 1-18)

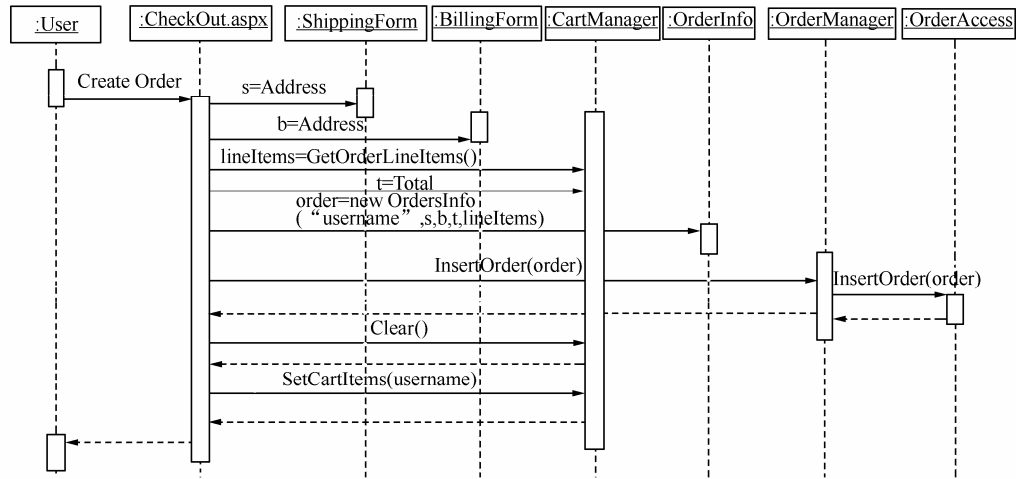


图 1-18 订单生成序列图

3. 订单页面 (见图 1-19)



图 1-19 订单页面

1.4.7 系统用户界面总览

1. 首页（见图 1-20）



图 1-20 首页

2. 图书概要页（见图 1-21）



图 1-21 图书概要页

3. 图书详情页（见图 1-22）



图 1-22 图书详情页

4. 购物车页（见图 1-23）



图 1-23 购物车页

5. 订单生成页（见图 1-24）



蓝星网上购书管理系统

今天是2009年3月28日 星期六

蓝星书店

欢迎您的到来! 书店地址: 长春市卫星路3278号[130033] 服务热线: 0431-84602444

设为首页

关于我们

收藏本站

联系方式

首页

图书检索

我的购物车

我的订单

修改个人资料

会员注册

后台管理

Exit

会员登录

欢迎用户 [u]

使用长春职业技术学院网上书店

目录导航

- 计算机
- 文学类
- 建筑类
- 汽车类
- 旅游类
- 食品类
- 武侠类

真实姓名

所在国家

省份

所在城市

帐单地址

邮编

接收地址

Email

联系电话(8位)

下一步

版权所有: 长春职业技术学院软件技术专业 网站备案: 吉ICP备05002015号
学院地址: 长春市卫星路3278号[130033] 招生热线: 0431-84602444 84602555 84602666
请用IE5.0以上版本 1024*768以上分辨率浏览

图 1-24 订单生成页

6. 我的订单页（见图 1-25）



蓝星网上购书管理系统

今天是2009年3月28日 星期六

蓝星书店

欢迎您的到来! 书店地址: 长春市卫星路3278号[130033]

设为首页

关于我们

收藏本站

联系方式

首页

图书检索

我的购物车

我的订单

修改个人资料

会员注册

后台管理

Exit

会员登录

欢迎用户 [u]

使用长春职业技术学院网上书店

目录导航

- 计算机
- 文学类
- 建筑类
- 汽车类
- 旅游类
- 食品类
- 武侠类

当前所在位置: >> 首页 > 我的订单

订单明细: 请选择订单列表查看相应明细

明细编号	图书编号	单价
1	9	41.00
2	11	65.00

订单列表:

选择	订单编号	帐单地址	接收地址	订单总价
选择	36	长春人民大街	长春卫星路	82.00
选择	37	长春人民大街	长春卫星路	115.00
选择	38	长春人民大街	长春卫星路	106.00

版权所有: 长春职业技术学院软件技术专业 网站备案: 吉ICP备05002015号
学院地址: 长春市卫星路3278号[130033] 招生热线: 0431-84602444 84602555 84602666
请用IE5.0以上版本 1024*768以上分辨率浏览

图 1-25 我的订单页

• 30 •

7. 图书检索页（见图 1-26）



图 1-26 图书检索页

1.4.8 数据库设计

如在任务分析中所述，面向对象的数据库设计是从对象模型出发的，属于实体主导型设计。我们由如图 1-27 所示的业务实体出发，完成向数据库概念模型的映射。

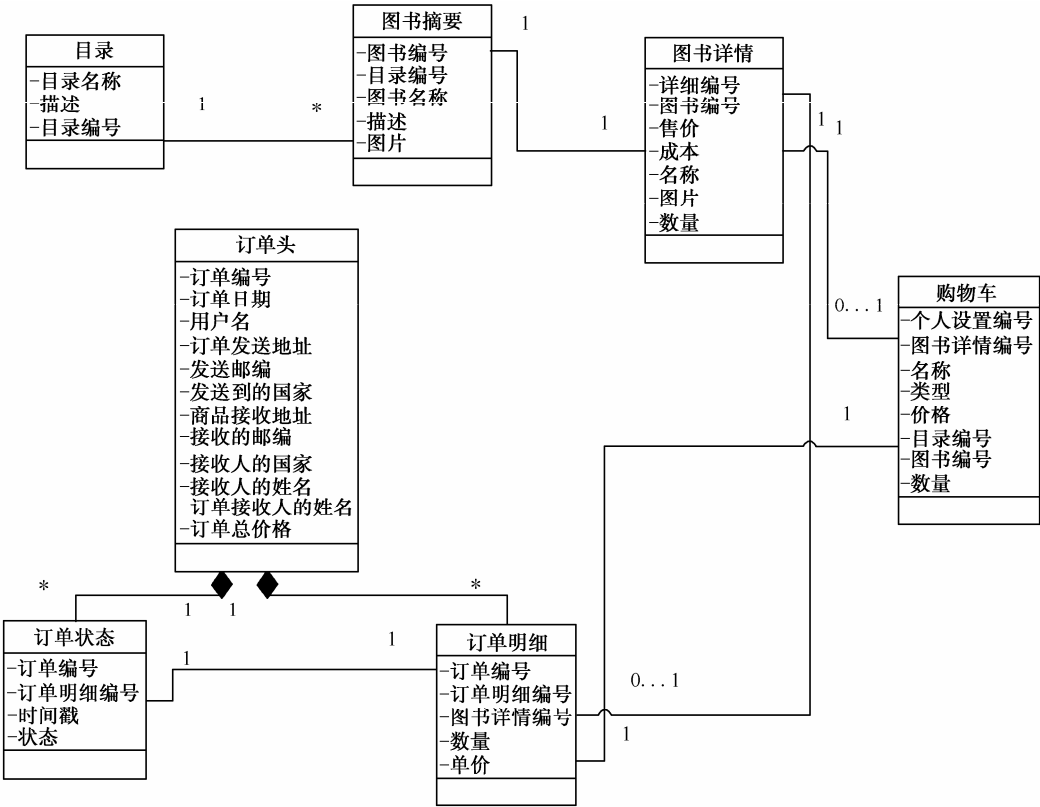


图 1-27 业务实体类

1. 表的设计

(1) 表的概述

① 用户自定义的数据表，见表 1-18。

表 1-18 用户自定义的数据表

序 号	表 名	含 义	初 始 大 小	最大增长范围	增 长 方 式
1	Category	图书类别表	128KB	640KB	自动计算，增长速率：1 行/月
2	BookBrief	图书概要信息表	256KB	16MB	自动计算，增长速率：20 行/月
3	Item	图书详情信息表	256KB	16MB	自动计算，增长速率：20 行/月
4	Cart	购物车信息表	128KB	10MB	自动计算，增长速率：1 000 行/月
5	Orders	订单头信息表	256KB	51MB	自动计算，增长速率：30 行/月
6	LineItem	订单明细表	128KB	16MB	自动计算，增长速率：1 000 行/月
7	OrderStatus	订单状态表	128KB	16MB	自动计算，增长速率：1 000 行/月

② 除上述用户自定义的数据表之外，还应用了 ASP.NET 自动生成的数据表，这部分数据表的详情如表 1-19 所示。

表 1-19 ASP.NET 成员资格自动生成的数据表

表 名	说 明
aspnet_Applications	应用程序的基本信息：程序名、程序描述等
aspnet_Membership	用户的详细信息：用户名、邮箱等
aspnet_Paths	应用程序路径信息
aspnet_PersonalizationAllUser	存储所有用户的个性化信息
aspnet_PersonalizationPerUser	存储特定用户的个性化信息
aspnet_Profile	个性化配置的内容
aspnet_Roles	角色表
aspnet_SchemaVersions	各部分的版本信息
aspnet_Users	用户表
aspnet_UsersInRoles	用户与角色的关系表
aspnet_WebEvent_Events	存储事件日志信息

(2) 表的详细设计

① Category 表，见表 1-20。

表 1-20 Category 表结构

字 段 名	类 型	默 认 值	索 引	PK/FK	允 许 空	说 明
CategoryId	smallint		Yes	PK	No	类别编号
CategoryName	nvarchar(50)		Yes		No	类别名称
Desc	nvarchar(50)		No		Yes	类别描述

② BookBrief 表，见表 1-21。

表 1-21 BookBrief 表结构

字 段 名	类 型	默 认 值	索 引	PK/FK	允 许 空	说 明
BookId	varchar(10)		Yes	PK	No	图书编号
CategoryId	smallint		Yes	FK	No	目录编号
BookName	varchar(80)		No		No	图书名称
BookDesc	varchar(255)		No		No	图书描述
BookImage	varchar(80)		No		No	图书图片
AddTime	datetime	getdate()	No		No	添加时间
IsHeadLine	bit		No		No	是否热点推荐
IsClassic	bit		No		No	是否经典书目

③ Item 表，见表 1-22。

表 1-22 Item 表结构

字 段 名	类 型	默 认 值	索 引	PK/FK	允 许 空	说 明
BookId	varchar(10)		Yes	PK/FK	No	图书编号
BookAuthor	nvarchar(50)		No		No	作者
Publisher	nvarchar(50)		No		No	出版社
PublishDate	datetime		No		No	出版日期
BookPrice	decimal(10,2)		No		No	图书市价
ListPrice	decimal(10,2)		No		No	售价
UnitCost	decimal(10,2)		No		No	成本价
ItemName	varchar(80)		No		No	详细书名
ItemImage	varchar(80)		No		No	详细图片
Qty	int	0	No		No	数量

④ Cart 表，见表 1-23。

表 1-23 Cart 表结构

字 段 名	类 型	默 认 值	索 引	PK/FK	允 许 空	说 明
UserName	varchar(256)		Yes	PK	No	用户名
BookId	varchar(10)		Yes	PK/FK	No	图书编号
Price	decimal(10,2)		No		No	售价
ItemName	varchar(80)		No		No	详细书名
Quantity	int	0	No		No	数量

⑤ Orders 表，见表 1-24。

表 1-24 Orders 表结构

字 段 名	类 型	默 认 值	索 引	PK/FK	允 许 空	说 明
OrderId	int		Yes	PK	No	订单编号
UserId	varchar(20)		No		No	用户名
OrderDate	datetime	getdate()	No		No	订单日期
ShipToName	varchar(80)		No		No	接货人姓名
ShipEmail	varchar(80)		No		No	接货人电邮
ShipAddr	varchar(80)		No		No	接货人地址
ShipCity	varchar(80)		No		No	接货人所在城市
ShipState	varchar(80)		No		No	接货人所在省份
ShipZip	varchar(20)		No		No	接货人邮编
ShipCountry	varchar(20)		No		No	接货人所在国家
ShipPhone	varchar(20)		No		No	接货人电话
BillToName	varchar(80)		No		No	付款人姓名
BillEmail	varchar(80)		No		No	付款人电邮
BillAddr	varchar(80)		No		No	付款人地址
BillCity	varchar(80)		No		No	付款人所在城市
BillState	varchar(80)		No		No	付款人所在省份
BillZip	varchar(20)		No		No	付款人邮编
BillCountry	varchar(20)		No		No	付款人所在国家
BillPhone	varchar(20)		No		No	付款人电话
TotalPrice	decimal(10,2)		No		No	数量

⑥ LineItem 表，见表 1-25。

表 1-25 LineItem 表结构

字 段 名	类 型	默 认 值	索 引	PK/FK	允 许 空	说 明
OrderId	int		Yes	PK	No	订单编号
LineNum	int		Yes	PK	No	订单明细号
BookId	varchar(10)		No		No	图书编号
Quantity	int		No		No	数量
UnitPrice	decimal(10,2)		No		No	售出单价

⑦ OrderStatus 表，见表 1-26。

表 1-26 OrderStatus 表结构

字 段 名	类 型	默 认 值	索 引	PK/FK	允 许 空	说 明
OrderId	int		Yes	PK	No	订单编号
LineNum	int		Yes	PK	No	订单明细号
Timestamp	datetime	getdate()	No		No	时间戳
Status	varchar(2)		No		No	订单状态

(3) 数据库关系图

数据库关系图如图 1-28 所示。

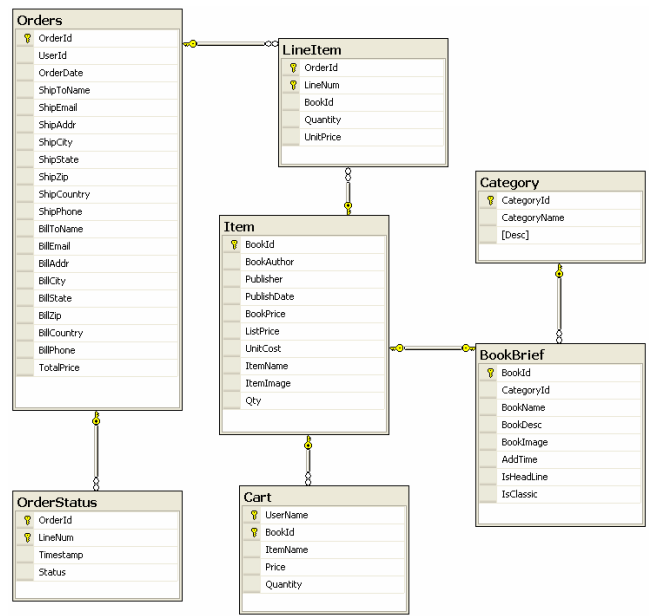


图 1-28 数据库关系图

2. 数据库备份与恢复说明

系统安装配置后，采取定期海量数据备份与增量备份相结合的备份方案。其中，海量数据备份频率为每周；增量数据备份频率为每天。

当数据库数据由于各种特殊原因而被破坏时，可以分以下三步进行恢复：

- ① 先取上一周的完整备份数据进行数据库的恢复；
- ② 再取每天的增量备份数据恢复到上一天结束时的运行状态；
- ③ 使用日志文件，恢复数据到破坏前状态。

3. 空间分配

数据库初始空间分配为 1.5MB，最大空间分配为 136MB。

共享池为 300~500MB。数据缓冲区应尽可能地大。应用服务器的缓冲区设定为 500MB。

4. 冗余说明

为保障数据安全，数据库选用 RAID 5 技术规范。

1.4.9 任务小结

通过本任务的学习，完成了子系统的设计，主要为针对类的设计、业务服务实现设计、用户界面设计及数据库设计，其中能够理解与灵活应用面向对象程序思想及 RUP 统一开发过程是完成本任务的关键。

1.4.10 练习题

- 1. 如何理解用例驱动？
- 2. 类的设计原则有哪些？

1.5 开发前期的解决方案构建

学习目标

- 掌握在 VS2008 中创建包含多个项目的解决方案
- 掌握母版页、主题及站点地图的应用
- 掌握业务实体类的实现
- 建议学时：16 学时

1.5.1 任务名称：开发前期的解决方案构建

1.5.2 任务描述

创建解决方案：本项目解决方案由 5 个项目组成，分别是网站 Web、类库 BLL、类库 DAL、类库 Model 和类库 Common。

- ① 在类库 Model 中创建业务实体类。
- ② 在类库 Common 中创建公共类数据库连接类 DataBase。
- ③ 创建网站母版页及设计主题。
- ④ 创建站点地图。

1.5.3 任务分析

网站 Web 对应表示层，主要应包含网页页面、用户控件等，类库 BLL 中包含所有业务逻辑层中的类，类库 DAL 中包含所有数据访问层中的类，类库 Model 中则包含所有业务实体类，类库 Common 包含公共类。各项目的详细内容可参见第 4 节子系统设计中的类的列表。如果一个类库中的类要调用另一个类库中的类，则要在该类库中添加对另一个类库的引用。

在设计母版页时则要注意，母版页是一个页面模板，包含的是页面的公共部分。因此，在创建母版页之前，必须判断哪些内容是页面的公共部分，比较常见的公共部分通常有片头、页脚、功能侧栏等。单独的母版页只是一个页面模板，它不能在浏览器中被打开，只有将其应用到具体的某个内容页上，在浏览器中访问该内容页，其才能发挥作用。

站点地图的扩展名为.sitemap，是 ASP 2.0 及 ASP 3.5 提供的站点导航控件——如本项目中使用的 SiteMapPath 控件提供站点层次结构信息的标准 XML 文件。

1.5.4 创建解决方案

① 运行 VS2008，在菜单栏选择“文件”→“新建”→“项目”，在“新建项目”对话框“项目类型”中展开“其他项目类型”节点，选择“Visual Studio 解决方案”；在模板窗格使用默认的“空白解决方案”模板，选定适当的位置，创建名称为“BlueStarBookShop”的解决方案，如图 1-29 所示。

② 右击新建的“解决方案‘BlueStarBookShop’（0 个项目）”，在“添加”菜单项的展开菜单中选择“新建网站”，如图 1-30 所示，创建网站 Web。

创建的网站 Web 如图 1-31 所示。

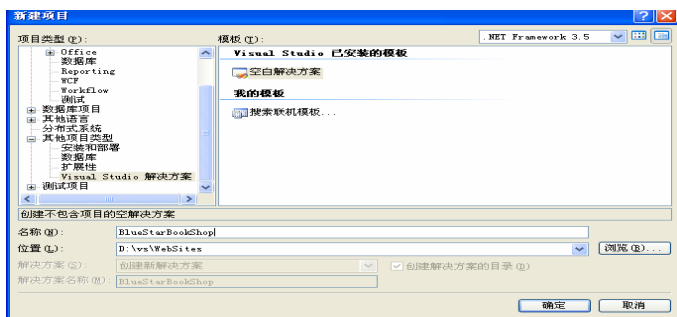


图 1-29 创建空白解决方案



图 1-30 在解决方案中新建网站

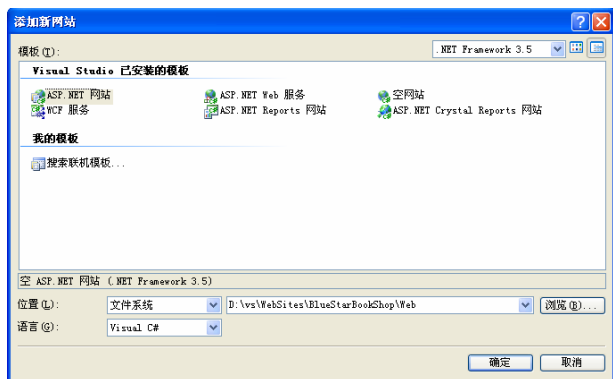


图 1-31 创建网站 Web

③ 在解决方案中添加一个网站项目之后，继续右击新建的“解决方案‘BlueStarBookShop’（1 个项目）”，在“添加”菜单项的展开菜单中选择“新建项目”，如图 1-32 所示。在“添加新项目”对话框的“模板”窗格中选择“类库”选项，如图 1-33 所示，在解决方案中创建类库 BLL。

重复第 3 步的步骤，继续在解决方案中依次创建类库 DAL、类库 Model、类库 Common。创建完成的解决方案如图 1-34 所示。

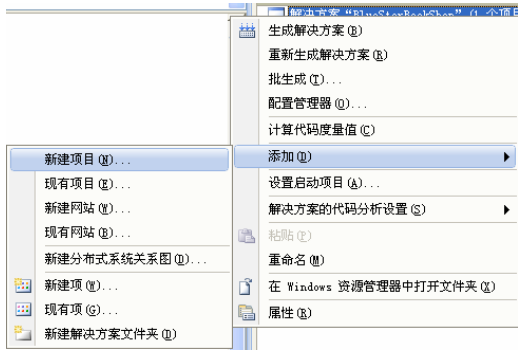


图 1-32 新建项目

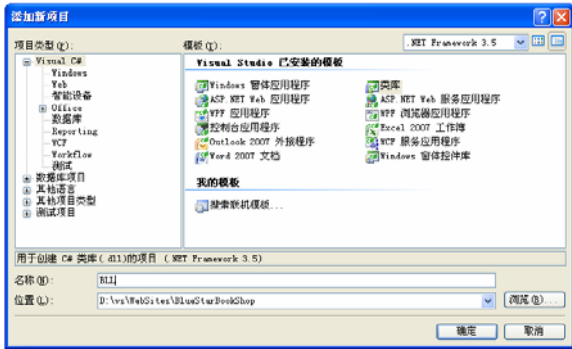


图 1-33 创建类库 BLL

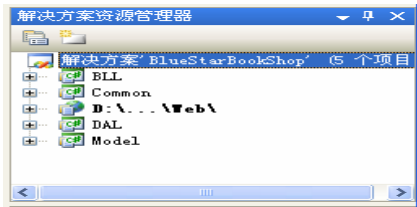


图 1-34 完整的解决方案

1.5.5 在类库Model中创建业务实体类

1. 图书目录实体类CategoryInfo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace BookShop.Model
{
    [Serializable]
    public class CategoryInfo
    {
```

```

        private Int16 id;//目录编号
        public Int16 Id
        {
            get { return id; }
            set { id = value; }
        }

        private string name;//目录名称
        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        private string description;//目录描述
        public string Description
        {
            get { return description; }
            set { description = value; }
        }
    }
}

```

2. 图书概要实体类BookBriefInfo

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace BookShop.Model
{
    /// <summary>
    /// Business entity used to model a product
    /// </summary>
    [Serializable]
    public class BookBriefInfo {
        // Internal member variables
        private string bookId;//书号
        public string BookId
        {
            get { return bookId; }
            set { bookId = value; }
        }

        private string bookName;//图书概要名称
        public string BookName
        {

```

```

        get { return bookName; }
        set { bookName = value; }
    }

    private string bookDesc;//图书介绍
    public string BookDesc
    {
        get { return bookDesc; }
        set { bookDesc = value; }
    }

    private string bookImage;//图书概要图片
    public string BookImage
    {
        get { return bookImage; }
        set { bookImage = value; }
    }

    private string categoryId;//目录编号
    public string CategoryId
    {
        get { return categoryId; }
        set { categoryId = value; }
    }

    private DateTime addTime;//添加时间
    public DateTime AddTime
    {
        get { return addTime; }
        set { addTime = value; }
    }

    private bool isHeadLine;//热点
    public bool IsHeadLine
    {
        get { return isHeadLine; }
        set { isHeadLine = value; }
    }

    private bool isClassic;//经典
    public bool IsClassic
    {
        get { return isClassic; }
        set { isClassic = value; }
    }
}

/// <summary>
/// Default constructor
/// </summary>

```



```

        public BookBriefInfo() { }
        /// <summary>
        /// Constructor with specified initial values
        /// </summary>
        public BookBriefInfo(string id, string name, string description, string image, string categoryId,
DateTime addtime,bool headline,bool classic) {
            this.bookId = id;
            this.bookName = name;
            this.bookDesc = description;
            this.bookImage = image;
            this.categoryId = categoryId;
            this.addTime = addtime;
            this.isHeadLine = headline;
            this.isClassic = classic;
        }
    }
}

```

3. 图书详情实体类ItemInfo

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BookShop.Model
{
    [Serializable]
    public class ItemInfo
    {
        string bookId;
        public string BookId
        {
            get { return bookId; }
            set { bookId = value; }
        }

        string bookAuthor;
        public string BookAuthor
        {
            get { return bookAuthor; }
            set { bookAuthor = value; }
        }

        string publisher;
        public string Publisher
        {

```

```

        get { return publisher; }
        set { publisher = value; }
    }

    DateTime publishDate;
    public DateTime PublishDate
    {
        get { return publishDate; }
        set { publishDate = value; }
    }

    decimal bookPrice;
    public decimal BookPrice
    {
        get { return bookPrice; }
        set { bookPrice = value; }
    }

    decimal listPrice;
    public decimal ListPrice
    {
        get { return listPrice; }
        set { listPrice = value; }
    }

    decimal unitCost;
    public decimal UnitCost
    {
        get { return unitCost; }
        set { unitCost = value; }
    }

    string itemName;
    public string ItemName
    {
        get { return itemName; }
        set { itemName = value; }
    }

    string itemImage;
    public string ItemImage
    {
        get { return itemImage; }
        set { itemImage = value; }
    }

    int qty;
    public int Qty

```

```

    {
        get { return qty; }
        set { qty = value; }
    }
    /// <summary>
    /// 默认构造
    /// </summary>
    public ItemInfo() { }
    /// <summary>
    /// 带参数的构造
    /// </summary>
    /// <param name="BookId">书号</param>
    /// <param name="BookAuthor">作者</param>
    /// <param name="Publisher">出版社</param>
    /// <param name="PublishDate">出版日期</param>
    /// <param name="BookPrice">市场价</param>
    /// <param name="ListPrice">售价</param>
    /// <param name="UnitCost">成本价</param>
    /// <param name="ItemName">图书详细名称</param>
    /// <param name="ItemImage">图书详细图片</param>
    /// <param name="Qty"></param>
    public ItemInfo(string BookId,string BookAuthor,string Publisher,DateTime PublishDate, decimal
BookPrice,decimal ListPrice,decimal UnitCost,string ItemName,string ItemImage,int Qty)
    {
        this.bookId = BookId;
        this.bookAuthor = BookAuthor;
        this.publisher = Publisher;
        this.publishDate = PublishDate;
        this.bookPrice = BookPrice;
        this.listPrice = ListPrice;
        this.unitCost = UnitCost;
        this.itemName = ItemName;
        this.itemImage = ItemImage;
        this.qty = Qty;
    }
}
}

```

4. 购物车实体类CartInfo

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BookShop.Model
{

```

```

[Serializable]
public class CartInfo
{
    string bookId;//书号
    public string BookId
    {
        get { return bookId; }
        set { bookId = value; }
    }

    string itemName;//图书详细名称
    public string ItemName
    {
        get { return itemName; }
        set { itemName = value; }
    }

    decimal price;//售价
    public decimal Price
    {
        get { return price; }
        set { price = value; }
    }

    int quantity;//数量
    public int Quantity
    {
        get { return quantity; }
        set { quantity = value; }
    }
    /// <summary>
    /// 默认构造
    /// </summary>
    public CartInfo() { }
    /// <summary>
    /// 带参数的构造
    /// </summary>
    /// <param name="bookId"></param>
    /// <param name="itemName"></param>
    /// <param name="price"></param>
    /// <param name="quantity"></param>
    public CartInfo(string bookId, string itemName, decimal price, int quantity)
    {
        this.bookId = bookId;
        this.itemName = itemName;
        this.price = price;
        this.quantity = quantity;
    }
}

```

```
    }  
    }  
}
```

5. 订单类OrdersInfo

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace BookShop.Model  
{  
    [Serializable]  
    public class OrdersInfo  
    {  
        //内部成员变量  
        private string userId;//用户标识  
        private AddressInfo shippingAddress;//订单发送地址  
        private AddressInfo billingAddress;//货物接收地址  
        private decimal totalPrice;//订单总价格  
        private LineItemInfo[] lineItems;//订单明细  
  
        public string UserId  
        {  
            get { return userId; }  
            set { userId = value; }  
        }  
        public AddressInfo ShippingAddress  
        {  
            get { return shippingAddress; }  
            set { shippingAddress = value; }  
        }  
        public AddressInfo BillingAddress  
        {  
            get { return billingAddress; }  
            set { billingAddress = value; }  
        }  
        public decimal TotalPrice  
        {  
            get { return totalPrice; }  
            set { totalPrice = value; }  
        }  
        public LineItemInfo[] LineItems  
        {  
            get { return lineItems; }  
            set { lineItems = value; }  
        }  
    }  
}
```

```

    }

    //默认的构造方法
    public OrdersInfo() { }

    //带参数的构造方法

    public OrdersInfo(string userId, AddressInfo shippingAddress, AddressInfo billingAddress,
decimal totalPrice, LineItemInfo[] lineItems)
    {
        this.userId = userId;
        this.shippingAddress = shippingAddress;
        this.billingAddress = billingAddress;
        this.totalPrice = totalPrice;
        this.lineItems = lineItems;
    }
}
}

```

6. 订单明细类LineItemInfo

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BookShop.Model
{
    [Serializable]
    public class LineItemInfo
    {
        int lineNum;
        public int LineNum
        {
            get { return lineNum; }
            set { lineNum = value; }
        }

        string bookId;
        public string BookId
        {
            get { return bookId; }
            set { bookId = value; }
        }

        int quantity;
        public int Quantity
        {

```

```

        get { return quantity; }
        set { quantity = value; }
    }

    decimal unitPrice;
    public decimal UnitPrice
    {
        get { return unitPrice; }
        set { unitPrice = value; }
    }
    //不带参数的构造函数
    public LineItemInfo() { }
    /// <summary>
    /// 带参数的构造函数
    /// </summary>
    /// <param name="lineNum">明细编号</param>
    /// <param name="bookId">书号</param>
    /// <param name="quantity">数量</param>
    /// <param name="unitPrice">价格</param>
    public LineItemInfo(int lineNum, string bookId, int quantity, decimal unitPrice)
    {
        this.lineNum = lineNum;
        this.bookId = bookId;
        this.quantity = quantity;
        this.unitPrice = unitPrice;
    }
    public decimal Subtotal
    {
        get
        {return this.unitPrice*this.quantity;}
    }
}
}

```

7. 地址类AddressInfo

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BookShop.Model
{
    [Serializable]
    public class AddressInfo
    {
        string email;//电子邮件
    }
}

```

```
public string Email
{
    get { return email; }
    set { email = value; }
}

string name;//真实姓名
public string Name
{
    get { return name; }
    set { name = value; }
}

string address;//地址
public string Address
{
    get { return address; }
    set { address = value; }
}

string city;//城市
public string City
{
    get { return city; }
    set { city = value; }
}

string state;//省份
public string State
{
    get { return state; }
    set { state = value; }
}

string zip;//邮编
public string Zip
{
    get { return zip; }
    set { zip = value; }
}

string country;//国家
public string Country
{
    get { return country; }
    set { country = value; }
}
```



```

        string phone;//电话
        public string Phone
        {
            get { return phone; }
            set { phone = value; }
        }
        /// <summary>
        /// 默认构造
        /// </summary>
        public AddressInfo() { }
        /// <summary>
        /// 带参数的构造
        /// </summary>
        public AddressInfo(string email, string name, string address, string city, string state, string zip,
string country, string phone)
        {
            this.email = email;
            this.name = name;
            this.address = address;
            this.city = city;
            this.state = state;
            this.zip = zip;
            this.country = country;
            this.phone = phone;
        }
    }
}

```

1.5.6 在类库Common中创建公共类数据库连接类DataBase

```

using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Xml.Linq;
using System.Data.SqlClient;

/// <summary>
/// DataBase 的摘要说明
/// </summary>
namespace BookShop.Common
{
    public class DataBase
    {

```

```

string connectString;
SqlConnection connection;
public DataBase()
{
    this.connectString = ConfigurationSettings.AppSettings["conn"];
}
public void Open()
{
    if (this.connection == null)
    {
        this.connection = new SqlConnection(this.connectString); //建立一个连接
        this.connection.Open();
    }
    //如果连接状态是关闭的
    if (this.connection.State.Equals(ConnectionState.Closed))
    {
        this.connection.Open();
    }
}

public void Close()
{
    if (this.connection != null) //如果连接存在
    {
        this.connection.Close();
    }
}

public SqlConnection GetConnection()
{
    this.Open();
    return connection;
}

```

```

/// <summary>
/// 对 SqlCommand 属性作设置
/// </summary>
/// <param name="cmd">SqlCommand 对象</param>
/// <param name="ct">SqlCommand 对象的命令类型</param>
/// <param name="cmdTxt">SqlCommand 对象的文本</param>
/// <param name="cmdParms">SqlCommand 对象的参数</param>

```

```

public void Preparecommand(SqlCommand cmd, CommandType ct, string cmdTxt, Sql
Parameter[] cmdParms)
{
    this.Open(); //打开连接
    cmd.Connection = this.connection; //指明这个 Command 是基于已打开的这个连接
    cmd.CommandText = cmdTxt;
    cmd.CommandType = ct;
}

```

```

        if (cmdParms != null)//如果使用存储过程
        {
            foreach (SqlParameter parm in cmdParms)
            {
                cmd.Parameters.Add(parm);
            }
        }
    }

    public void Preparecommand(SqlCommand cmd, SqlConnection conn, SqlTransaction trans,
    CommandType ct, string cmdTxt, SqlParameter[] cmdParms)
    {
        //this.Open();
        cmd.Connection = conn;
        cmd.Transaction = trans;//把事务和 command 关联
        cmd.CommandText = cmdTxt;
        cmd.CommandType = ct;
        if (cmdParms != null)//如果使用存储过程
        {
            foreach (SqlParameter parm in cmdParms)
            {
                cmd.Parameters.Add(parm);
            }
        }
    }
    /// <summary>
    /// 对数据库进行增删改方法
    /// </summary>
    /// <param name="ct">SqlCommand 对象的命令类型</param>
    /// <param name="cmdTxt">SqlCommand 对象的文本</param>
    /// <param name="cmdParms">SqlCommand 对象的参数</param>
    ///
    public void ExcuteNonQuery(CommandType ct, string cmdTxt, SqlParameter[] cmdParms)
    //对数据库进行增删改方法
    {
        SqlCommand cmd = new SqlCommand();
        this.Preparecommand(cmd, ct, cmdTxt, cmdParms);
        cmd.ExecuteNonQuery();
        cmd.Parameters.Clear();
        this.Close();
    }

    //对数据库进行增删改方法（使用事务）
    public void ExcuteNonQuery(SqlTransaction trans,CommandType ct, string cmdTxt,
    SqlParameter[] cmdParms)
    {

```

```

        SqlCommand cmd = new SqlCommand();
        this.Preparecommand(cmd,trans.Connection,trans.ct, cmdTxt, cmdParms);
        cmd.ExecuteNonQuery();
        cmd.Parameters.Clear();
    }

    //对数据库进行增删改方法
    public object ExcuteNonQueryReturn(SqlTransaction trans,CommandType ct, string cmdTxt,
    SqlParameter[] cmdParms)
    {
        SqlCommand cmd = new SqlCommand();
        this.Preparecommand(cmd,trans.Connection,trans, ct, cmdTxt, cmdParms);
        cmd.ExecuteNonQuery();
        object ob = cmd.Parameters["@Return"].Value;//获取数组中名为"@Return"的值(orderId)
        return ob;
    }

    /// <summary>
    /// 读取数据
    /// </summary>
    /// <param name="ct">SqlCommand 对象的命令类型</param>
    /// <param name="cmdTxt">SqlCommand 对象的文本</param>
    /// <param name="cmdParms">SqlCommand 对象的参数</param>
    /// <returns></returns>
    public SqlDataReader ExcuteDataReader(CommandType ct, string cmdTxt, SqlParameter[]
    cmdParms)//读取数据
    {

        SqlCommand cmd = new SqlCommand();
        this.Preparecommand(cmd, ct, cmdTxt, cmdParms);
        return cmd.ExecuteReader(CommandBehavior.CloseConnection);
    }

    //调用名为 cmdText 的存储过程
    public SqlDataReader ExecuteDataReader(string cmdText)
    {
        SqlCommand cmd = new SqlCommand();
        SqlConnection conn = new SqlConnection(connectString);
        try
        {
            this.Preparecommand(cmd, conn, null, CommandType.StoredProcedure, cmdText,
            null);

            SqlDataReader rdr = cmd.ExecuteReader();
            cmd.Parameters.Clear();
            return rdr;
        }
    }

```

```

        catch
        {
            conn.Close();
            throw;
        }
    }

    public object ExcuteScalar(CommandType ct, string cmdTxt, SqlParameter[] cmdParms)
//返回第一行第一列的一个数值
    {
        SqlCommand cmd = new SqlCommand();
        this.Preparecommand(cmd, ct, cmdTxt, cmdParms);
        return cmd.ExecuteScalar();
    }

    /// <summary>
    /// 创建 DataAdapter 方法
    /// </summary>
    /// <param name="ct">SqlCommand 的命令类型</param>
    /// <param name="cmdTxt">SqlCommand 对象的文本</param>
    /// <param name="cmdParms">SqlCommand 对象的参数</param>
    /// <param name="opt">整型数值</param>
    /// <returns></returns>
    public SqlDataAdapter CreateDataAdapter(CommandType ct, string cmdTxt, SqlParameter[]
cmdParms, int opt)
    {
        SqlDataAdapter sda = new SqlDataAdapter();
        SqlCommand cmd = new SqlCommand();
        this.Preparecommand(cmd, ct, cmdTxt, cmdParms);
        switch (opt)
        {
            case 1:
                sda.SelectCommand = cmd;
                break;
            case 2:
                sda.InsertCommand = cmd;
                break;
            case 3:
                sda.DeleteCommand = cmd;
                break;
            case 4:
                sda.UpdateCommand = cmd;
                break;
            default:
                break;
        }
        this.Close();
    }

```

```
        return sda;
    }
}
}
```

1.5.7 设计解决方案中网站Web的母版页

本项目母版页由片头、导航栏、滚动栏、侧栏、页脚组成。

1. 母版页效果图

母版页组成如图 1-35 所示。



图 1-35 母版页组成

2. 母版页面中各个控件的属性设置及其用途

母版页面中各个控件的属性设置及其用途如表 1-27 所示。

表 1-27 母版页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
Label	lblDate	Text 属性设置为 “ ”	显示当前系统日期
Repeater	repCat		显示图书类别信息
TextBox	txtUserName		用户名输入框
	txtPassword		密码输入框
RequiredFieldValidator	rfvName	ErrorMessage 属性设置为 “用户名不能为空”	验证用户名不能为空
	rfvPwd	ErrorMessage 属性设置为 “密码不能为空”	验证密码不能为空
ImageButton	ibtnLogin	Text 属性设置为 “登录”	执行计算购物车总价格的操作
	ibtnResi	Text 属性设置为 “注册”	执行清空购物车所有图书的操作
Placeholder	pgLogin		实现动态加载控件, 用户登录前显示登录框, 登录后显示欢迎信息

控 件 类 型	控 件 名 称	主要属性设置	用 途
HyperLink	hlDefault	NavigateUrl 属性设置为 “~/Default.aspx”	链接到 “首页”
	hlSearch	NavigateUrl 属性设置为 “~/Default.aspx”	链接到 “图书检索页”
	hlCart	NavigateUrl 属性设置为 “~/ShoppingCart.aspx”	链接到 “我的购物车页”
	hlOrder	NavigateUrl 属性设置为 “~/MyOrder.aspx”	链接到 “我的订单页”
	hlProfile	NavigateUrl 属性设置为 “~/UserProfiles.aspx”	链接到 “修改个人资料页”
	hlRegister	NavigateUrl 属性设置为 “~/Register.aspx”	链接到 “会员注册页”
	hlAdmin	NavigateUrl 属性设置为 “~/admin/Default.aspx”	链接到 “后台管理首页”

步骤:

- ① 右击网站 Web，选择“添加新项”，在弹出的对话框中选择“母版页”模板，单击“确定”按钮新建母版页 MasterPage.master。
- ② 源视图 HTML 代码如下。

```
<% @ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
<% @ Register src="UserCtrl/PS.ascx" tagname="PS" tagprefix="uc1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>无标题页</title>

<%-- 左侧 div 与右侧 div 高度自适应脚本--%>
<script type="text/javascript">
window.onload=window.onresize=function(){
if(document.getElementById("left").clientHeight<document.getElementById("right").clientHeight){do
cument.getElementById("left").style.height=document.getElementById("right").offsetHeight+"px";}
else{
document.getElementById("right").style.height=document.getElementById("left").offsetHeight+"px";}
}
</script>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
<div id="total">
    <%--片头--%>
<div id="header"> </div>
    <%--导航栏--%>
    <div id="menu">
        <asp:Label ID="lblDate" runat="server"></asp:Label>
        <span style="margin-left:150px;">
            <asp:HyperLink ID="hlDefault" runat="server" Font-Bold="True" Navigate
```

```

Url="~/Default.aspx" >首页</asp:HyperLink>
        <asp:HyperLink ID="hlSearch" runat="server" Font-Bold="True" NavigateUrl=
"~/Search.aspx" >图书检索</asp:HyperLink>
        <asp:HyperLink ID="hlCart" runat="server" Font-Bold="True" NavigateUrl=
"~/ShoppingCart.aspx" >我的购物车</asp:HyperLink>
        <asp:HyperLink ID="hlOrder" runat="server" Font-Bold="True" NavigateUrl=
"~/MyOrder.aspx" >我的订单</asp:HyperLink>
        <asp:HyperLink ID="hlProfile" runat="server" Font-Bold="True" NavigateUrl=
"~/UserProfiles.aspx" >修改个人资料</asp:HyperLink>
        <asp:HyperLink ID="hlRegister" runat="server" Font-Bold="True" NavigateUrl=
"~/Register.aspx" >会员注册</asp:HyperLink>
        <asp:HyperLink ID="hlAdmin" runat="server" NavigateUrl="~/admin/Default.aspx"
Font-Bold="True" >后台管理</asp:HyperLink>
        <asp:ImageButton ID="logout" runat="server" ImageUrl="~/images/exit.gif" Visible="false"
CausesValidation="False" onclick="logout_Click"/>
    </span>
</div>
<!--滚动栏-->
    <div id="mar">
        <marquee onmouseover=this.stop() onmouseout=this.start() align="left"><FONT color="white"><b>
蓝星书店<span style="margin-left:30px; color:white; font-size:small;">欢迎您的到来! 书店地址: 长春市卫星
路 3278 号[130033].服务热线:0431-84602444</span>
        </b></FONT></marquee>
    </div>
<!--侧栏-->
    <div id="left">
        <p align="center">
            
        </p>
        <p align="center">

<!-- 使用 Placeholder 控件实现动态加载控件-->
        <asp:Placeholder ID="pgLogin" runat="server" >
            <p align="center" style="font-size:small;" >
                用户:<asp:TextBox ID="txtUserName" runat="server" Width="100px"></asp:TextBox>
                <asp:RequiredFieldValidator ID="rfvName" runat="server"
                    ControlToValidate="UserName" ErrorMessage="用户名不能为空">*</asp:
RequiredFieldValidator> </p>
                <p align="center" style="font-size:small;">
                    密码:<asp:TextBox ID="txtPassword" runat="server" TextMode="Password" Width=
"100px"></asp:TextBox>
                <asp:RequiredFieldValidator ID="rfvPwd" runat="server"
                    ControlToValidate="Password" ErrorMessage="密码不能为空">*</asp:Required
FieldValidator> </p>
            <p align="center" >
                <asp:ImageButton ID="ibtnLogin" runat="server" ImageUrl="~/images/btn1.gif" onclick=
"ibtnLogin_Click" />

```



```

        <asp:ImageButton ID="ibtnResi" runat="server" ImageUrl="~/images/btn2.gif"
            CausesValidation="False" onclick="ibtnResi_Click" /> </p>
    </asp:PlaceHolder> </p>
    <p align="center">  </p>
    <div align="center">
    <%--显示图书类别信息--%>
        <asp:Repeater ID="repCat" runat="server" DataSourceID="sqlDsDirectory">
        <ItemTemplate>
            <li class="liclass" >
                <asp:HyperLink ID="hlDaoHang" runat="server" NavigateUrl='<%#~/BookBrief.
asp?CategoryId="+Eval("CategoryId")%>' Font-Underline="false" ><%# Eval("CategoryName")%></asp:Hyper
Link>
            </li>
        </ItemTemplate>
        </asp:Repeater>
    </div>
    <asp:SqlDataSource ID="sqlDsDirectory" runat="server"
        ConnectionString="<%= $ ConnectionStrings:CVITBOOKSHOPConnectionString2 %>"
        SelectCommand="SELECT * FROM [Category]"></asp:SqlDataSource>
    <asp:ValidationSummary ID="ValidationSummary1" runat="server" ShowMessageBox="True"
        ShowSummary="False" />
    </div>
    <%--工作区, ContentPlaceHolder 为母版页的占位符, 应用母版页的内容页的内容将被合并到该控
件中 --%>
    <div id="right">
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
    </div>
    <%--页脚--%>
    <div id="footer">
        <br /> 版权所有: 长春职业技术学院软件技术专业 网站备案: 吉 ICP 备 05002015 号

        <br /> 学院地址: 长春市卫星路 3278 号[130033] 招生热线: 0431-84602444 84602555 84602666
    <br />用 IE5.0 以上版本 1024*768 以上分辨率浏览
    </div>
    </div>
    </form>
</body>
</html>

```

1.5.8 MasterPage.master.cs代码实现

母版页中有登录框和登录注销按钮, 以及用来绑定图书类别信息的 Repeater 控件, 所以下面首要介绍.NET 身份验证及 Repeater 控件应用的相关知识。

1. 相关知识一: Forms身份验证凭据

FormsAuthentication 类: 为 Web 应用程序管理 Forms 身份验证服务, 用于在登录时验证用

户的 Forms 身份验证凭据，可以存储在外部数据源中，也可以存储在应用程序的配置文件中。

Web 应用程序通常采用 Forms 身份验证（还有 Windows 身份验证方式及 Passport 身份验证方式）进行用户和密码验证。使用 Forms 身份验证时，用户信息存储在外部数据源中，如 Membership 数据库，或存储在应用程序的配置文件中。在用户通过身份验证后，Forms 身份验证即会在 Cookie 或 URL 中维护一个身份验证票证，这样已通过身份验证的用户就无须在每次请求时都提供凭据。

可通过将 authentication 配置元素的 mode 属性设置为 Forms 来启用 Forms 身份验证。通过使用 authentication 配置元素可要求所有对应用程序的请求均需包含有效的用户身份验证票证，从而拒绝任何未知用户的请求。

本项目在 web.config 文件中将 authentication 配置元素的 mode 属性设置为 Forms，以此来启用 Forms 身份验证的代码。

```
<authentication mode="Forms">
  <forms name="BlueStarBookShop" loginUrl="Default.aspx" protection="None"/>
</authentication>
```

在本项目中设身份验证的默认页面为首页“Default.aspx”，由于网站前台页面允许匿名用户浏览图书信息，所以将保护 protection 属性设为“None”。对需要进行身份验证后才能访问的页面如购物车订单页面、修改会员个人资料页面等，采用增加 location 配置节来实现要求用户请求需包含有效的用户身份验证票证，从而拒绝任何未知用户的请求。在 Web.config 文件的<configuration></configuration>配置节中增加如下代码：

```
<location path="ShoppingCart.aspx">
  <system.web>
    <authorization>
      <deny users="?" />    //拒绝任何未知用户的请求
    </authorization>
  </system.web>
</location>

<location path="MyOrder.aspx">
  <system.web>
    <authorization>
      <deny users="?" />
    </authorization>
  </system.web>
</location>

<location path="CheckOut.aspx">
  <system.web>
    <authorization>
      <deny users="?" />
    </authorization>
  </system.web>
</location>
```

```
<location path="UserProfiles.aspx">
  <system.web>
    <authorization>
      <deny users="?"/>
    </authorization>
  </system.web>
</location>
```

通过以上配置当匿名用户要访问上述页面时就会跳转到首页，要求登录。

2. 相关知识二：ASP.NET成员资格

.NET 提供了一种验证和存储用户凭据的内置方法——ASP.NET 成员资格，它可帮助我们管理网站中的用户身份验证。

（1）ASP.NET 成员资格支持的功能

- ① 创建新用户和密码。
- ② 将成员资格信息（用户名、密码和支持数据）存储在 Microsoft SQL Server、Active Directory 或其他数据存储区。
- ③ 对访问站点的用户进行身份验证。可以以编程方式验证用户，也可以使用 ASP.NET 登录控件创建一个只需很少代码或无须代码的完整身份验证系统。
- ④ 管理密码，包括创建、更改和重置密码。根据选择的成员资格选项不同，成员资格系统还可以提供一个使用用户提供的问题和答案的自动密码重置系统。
- ⑤ 公开经过身份验证的用户的唯一标识，用户可以在自己的应用程序中使用该标识，也可以将该标识与 ASP.NET 个性化设置和角色管理（授权）系统集成。
- ⑥ 指定自定义成员资格提供程序，这使用户可以改为用自己的代码管理成员资格及在自定义数据存储区中维护成员资格数据。

（2）成员资格的工作原理

默认情况下，内置方法成员资格处于启用状态，并可以指定存储成员资格信息的数据库的类型。默认成员资格提供程序使用 Microsoft SQL Server 数据库。还可以选择使用 Active Directory 存储成员资格信息，或者可以指定自定义提供程序。

将应用程序配置为使用 Forms 身份验证，接着可以使用网站管理工具为成员资格创建用户账户。该工具提供了一个用于创建新用户的类似向导的界面。在 VS2008 平台下，选择菜单栏“网站”→“ASP.NET 配置”，将打开 ASP.NET 网站管理工具。在窗口中选择“安全”选项卡，将看到创建用户链接，如图 1-36 所示。单击“创建用户”，将打开创建新用户的类似向导的界面，如图 1-37 所示。如果在成员资格中启用角色，则也可以使用该工具创建角色，在新建用户的时候可同时指定它所属的角色。

创建好用户账户就可以登录，使用成员资格来进行身份验证了。在本项目中，母版页左侧栏的登录框用来完成用户登录。由于已将应用程序配置为使用 Forms 身份验证，因此当未经验证的用户请求一个受保护的页面时，ASP.NET 会自动显示登录页，在本项目中为首页。

使用成员资格类 Membership 类来完成成员资格管理，使用 Roles 类来管理角色中的用户成员资格，命名空间为 System.Web.Security。



图 1-36 “ASP.net 网站管理工具” / “安全”选项卡



图 1-37 创建成员资格新账户

Membership 类的方法如表 1-28 所示。

表 1-28 Membership 类的方法列表

名 称	说 明
CreateUser	已重载。将新用户添加到数据存储区
DeleteUser	已重载。从数据库中删除一个用户
FindUsersByEmail	已重载。获取一个成员资格用户的集合，其中的电子邮件地址包含要匹配的指定电子邮件地址
FindUsersByName	已重载。获取一个成员资格用户的集合，其中的用户名包含要匹配的指定用户名
GeneratePassword	生成指定长度的随机密码
GetAllUsers	已重载。获取数据库中用户的集合
GetNumberOfUsersOnline	获取当前访问应用程序的用户数
GetUser	已重载。从数据源获取成员资格用户的信息
GetUserNameByEmail	获取一个用户名，其中该用户的电子邮件地址与指定的电子邮件地址匹配
UpdateUser	用指定用户的信息更新数据库
ValidateUser	验证提供的用户名和密码是有效的

Membership 类的属性如表 1-29 所示。

表 1-29 Membership 类的属性列表

名 称	说 明
ApplicationName	获取或设置应用程序的名称
EnablePasswordReset	获得一个值，指示当前成员资格提供程序是否配置为允许用户重置其密码
EnablePasswordRetrieval	获得一个值，指示当前成员资格提供程序是否配置为允许用户检索其密码
HashAlgorithmType	用于哈希密码的算法的标识符
MaxInvalidPasswordAttempts	获取锁定成员资格用户前允许的无效密码或无效密码提示问题答案尝试次数
MinRequiredNonAlphanumericCharacters	获取有效密码中必须包含的最少特殊字符数
MinRequiredPasswordLength	获取密码所要求的最小长度
PasswordAttemptWindow	获取时间长度，在该时间间隔内对提供有效密码或密码答案的连续失败尝试次数进行跟踪
PasswordStrengthRegularExpression	获取用于计算密码的正则表达式
Provider	获取对应用程序的默认成员资格提供程序的引用
Providers	获取一个用于 ASP.NET 应用程序的成员资格提供程序的集合
RequiresQuestionAndAnswer	获取一个值，该值指示默认成员资格提供程序是否要求用户在进行密码重置和检索时回答密码提示问题
UserIsOnlineTimeWindow	指定用户在最近一次活动的日期/时间戳之后被视为联机的分钟数

Membership 类的事件如表 1-30 所示。

表 1-30 Membership 类的事件列表

名 称	说 明
ValidatingPassword	在创建用户、更改密码或重置密码时发生

关于 Roles 类将在讲解“会员注册”时介绍。

3. FormsAuthentication类与Membership类在MasterPage.master.cs中的应用

```
//用户登录
protected void ibtnLogin_Click(object sender, ImageClickEventArgs e)
{
    if (Membership.ValidateUser(txtUserName.Text, txtPassword.Text))
    {
        FormsAuthentication.RedirectFromLoginPage(txtUserName.Text,false);
    }
}

//登录注销
protected void logout_Click(object sender, EventArgs e)
{
    FormsAuthentication.SignOut();
    Response.Redirect("Default.aspx");
}
```

```
}
```

用户登录时调用 Membership 类的 ValidateUser 方法验证提供的用户名和密码是否有效。如果通过验证,调用 FormsAuthentication 的 RedirectFromLoginPage 方法,将通过身份验证的用户重定向回最初请求的 URL,该方法第一个参数表示经过身份验证的用户名,第二个参数指示是否为该用户创建持久 Cookie。参数设置为 true, Cookie 就会持久保留,设为 false 则不保留。

用户登录注销时调用 FormsAuthentication 的 SignOut()方法,从浏览器删除 Forms 身份验证票证。

4. 成员资格信息(用户名、密码和支持数据)存储位置

.NET Framework 默认的成员资格提供程序为 AspNetSqlProvider,它将用户信息存储在.NET 自带的 SQL 数据库——ASPNETDB.MDF 中,如果要把用户信息存储在自定义的 SQL 数据库中,则需要在 web.config 文件<configuration></configuration>配置节中进行如下配置:

```
<connectionStrings>
  <add name="CVITBOOKSHOPConnectionString" connectionString="Data Source=.\SQLEXPRESS;
Initial Catalog=CVITBOOKSHOP;Integrated Security=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
<system.web>
  <membership defaultProvider="SqlProvider" userIsOnlineTimeWindow="10">

    <providers>
      <remove name="AspNetSqlProvider"/>
      <add name="SqlProvider"
        type="System.Web.Security.SqlMembershipProvider"
        connectionStringName=" CVITBOOKSHOPConnectionString "
        enablePasswordRetrieval="false"
        enablePasswordReset="true"
        requiresQuestionAndAnswer="true"
        passwordFormat="Hashed"
        applicationName="/" />
    </providers>

  </membership>
</system.web>
```

除此之外,还要运行 ASP.NET SQL Server 安装向导 aspnet_regsql.exe,如图 1-38 所示。该工具所在的路径为 c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\,在自定义的数据库中创建成员资格的数据表和存储过程等。

5. 相关知识三: Repeater 控件

Repeater 服务器控件是一个数据绑定容器控件,用于生成各个项的列表。使用模板定义网页上各个项的布局。当该页运行时,该控件为数据源中的每个项重复此布局。

可以使用它从页的任何可用数据中创建出自定义列表。因为 Repeater 控件不具备内置的呈现功能,所以用户必须通过创建模板为 Repeater 控件提供布局。当该页运行时,Repeater

控件依次通过数据源中的记录，并为每个记录呈现一个项。

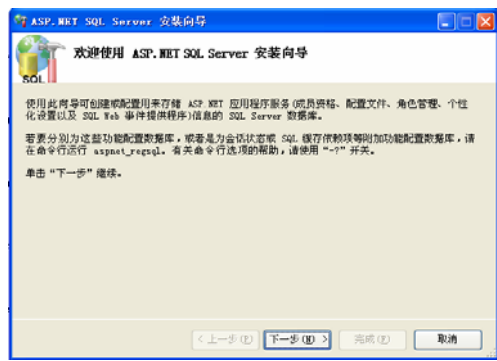


图 1-38 ASP.NET SQL Server 安装向导

若要使用 Repeater 控件，可创建定义控件内容布局的模板。模板可以包含标记和控件的任意组合。如果未定义模板，或者模板都不包含元素，则当应用程序运行时，该控件不显示在页上。

表 1-31 描述了 Repeater 控件支持的模板。

表 1-31 Repeater 控件支持的模板

模板 属性	说 明
ItemTemplate	包含要为数据源中每个数据项都呈现一次的 HTML 元素和控件
AlternatingItemTemplate	包含要为数据源中每个数据项都呈现一次的 HTML 元素和控件。通常，可以使用此模板为交替项创建不同的外观，如指定一种与在 ItemTemplate 中指定的颜色不同的背景色
HeaderTemplate 和 FooterTemplate	包含在列表的开始和结束处分别呈现的文本和控件
SeparatorTemplate	包含在每项之间呈现的元素。典型的示例可能是一条直线（使用 hr 元素）

使用时必须将 Repeater 控件绑定到数据源。最常用的数据源是数据源控件，如 SqlDataSource 或 ObjectDataSource 控件。或者，可以将 Repeater 控件绑定到任何实现 IEnumerable 接口的类，包括 ADO.NET 数据集（DataSet 类）、数据读取器（SqlDataReader 类或 OleDbDataReader 类）或大部分集合。

绑定数据时，可以为 Repeater 控件整体指定一个数据源。向 Repeater 控件添加控件时（例如，向模板中添加 Label 或 TextBox 控件时），可以使用数据绑定语法将单个控件绑定到数据源返回的项的某个字段。

6. 应用Repeater控件显示图书类别信息

在 Repeater 控件<ItemTemplate></ItemTemplate>模板中包含的为数据源中每个数据项都要呈现一次的 HTML 元素是列表项，在列表项中又包含 HyperLink 控件，其文本内容是绑定的数据源的“CategoryName”字段值，NavigateUrl 属性值为图书概要页 BookBrief.aspx，并向该页传递参数“CategoryId”，参数值为绑定的数据源的“CategoryId”字段值，用来实现按图书类别查看图书概要。

Repeater 控件的数据源是 SqlDataSource 控件，通过设置 Repeater 控件的“DataSourceID”属性值为“sqlDsDirectory”完成绑定。

HTML 码清单如下：

```
<--显示图书类别信息-->
```

```

<div align="center">
<asp:Repeater ID="repCat" runat="server" DataSourceID="sqlDsDirectory">
    <ItemTemplate>
        <li class="liclass">
            <asp:HyperLink ID="hlDaoHang" runat="server" NavigateUrl='<##~/BookBrief.aspx?
CategoryId="+Eval("CategoryId")%>' Font-Underline="false" ><## Eval("Category Name")%></asp:HyperLink>
        </li>
    </ItemTemplate>
</asp:Repeater>
</div>

<asp.SqlDataSource ID="sqlDsDirectory" runat="server" ConnectionString="<%%$ ConnectionStrings:
CVITBOOKSHOPConnectionString %>" SelectCommand="SELECT * FROM [Category]"></asp.SqlData
Source>

```

在 HTML 码清单中还使用了数据绑定表达式语法，数据绑定表达式包含在<##和%>分隔符之内，并使用 Eval 和 Bind 函数。Eval 函数用于定义单向（只读）绑定，Bind 函数用于定义双向（可更新）绑定。除了通过在数据绑定表达式中调用 Eval 和 Bind 方法执行数据绑定外，还可以调用<##和%>分隔符之内的任何公共范围代码，以在页面处理过程中执行该代码并返回一个值。

完整的 MasterPage.master.cs 代码清单如下：

```

using System;
using System.Configuration;
using System.Data;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class MasterPage : System.Web.UI.MasterPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string name=null;
        name = HttpContext.Current.User.Identity.Name; //获取通过身份验证的用户名
        if (!string.IsNullOrEmpty(name))
        {
            logout.Visible = true; //如果通过验证，显示注销按钮
            pgLogin.Controls.Clear(); //如果用户登录成功，清除掉 Placeholder 控件内原来加载的
            用户登录框
            Label lblshow = new Label(); //创建标签对象，显示通过身份验证后对用户的欢迎信息
            lblshow.ID = "lblshow";
            lblshow.Text = "欢迎用户[" + name + "]<br/>使用蓝星网上书店";
            lblshow.Font.Size = 8;
            lblshow.ForeColor = Color.Red;
            pgLogin.Controls.Add(lblshow); //加载到 Placeholder 控件,完成显示通过身份验证后对

```



```

    }
    else
        logout.Visible = false;
    string weekday = "";
    //判断当前系统日期是星期几
    switch (DateTime.Now.DayOfWeek.GetHashCode())
    {
        case 0:
            weekday = "星期日";
            break;
        case 1: weekday = "星期一";
            break;
        case 2: weekday = "星期二";
            break;
        case 3: weekday = "星期三";
            break;
        case 4: weekday = "星期四";
            break;
        case 5: weekday = "星期五";
            break;
        case 6: weekday = "星期六";
            break;
    }
    //获取当前系统日期
    lblDate.Text = "今天是" + DateTime.Now.ToLongDateString() + " " + weekday;
}
//登录注销
protected void logout_Click(object sender, EventArgs e)
{
    FormsAuthentication.SignOut();
    Response.Redirect("Default.aspx");
}
//用户登录
protected void ibtnLogin_Click(object sender, ImageClickEventArgs e)
{
    if (Membership.ValidateUser(UserName.Text, Password.Text))
    {
        FormsAuthentication.RedirectFromLoginPage(UserName.Text, false);
    }
}
//用户注册
protected void ibtnResi_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("Register.aspx");
}
}

```

1.5.9 为网站Web设置主题

主题是属性设置的集合可以用来定义页面和控件的外观，可以定义单个 Web 应用程序的主题，也可以定义供 Web 服务器上所有应用程序使用的全局主题。定义主题之后，可以使用 @ Page 指令的 Theme 或 StyleSheetTheme 属性将该主题放置在个别页上；或者通过设置应用程序配置文件中的 pages 元素（ASP.NET 设置架构），将其应用于应用程序中的所有页。如果在 Machine.config 文件中定义了 pages 元素（ASP.NET 设置架构），主题将应用于服务器上 Web 应用程序中的所有页。

主题由外观、级联样式表（CSS）、图像和其他资源元素组成，一个主题至少要包含外观文件。

外观文件的文件扩展名为.skin，它用来定义页面中服务器控件的外观。

级联样式表扩展名为.css，CSS 是 Cascading Style Sheets（层叠样式表）的简称，在标准网页设计中用来负责网页内容（XHTML）的表现。

主题还可以包含图形和其他资源，如脚本文件或声音文件。

1. 创建主题

① 在解决方案资源管理器中右击网站 Web，选择“添加 ASP.NET 文件夹”菜单项的子菜单项“主题”，如图 1-39 所示。

② 将本项目主题命名为 mytheme，右击该主题，选择“添加新项”，依次添加外观文件 SkinFile.skin 和样式文件 StyleSheet.css，如图 1-40 所示。



图 1-39 添加主题

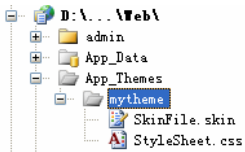


图 1-40 外观文件及样式文件

SkinFile.skin 文件代码示例如下：

```
<asp:HyperLink runat="server" Font-Size="Small" Font-Underline="false"></asp:HyperLink>
<asp:Label runat="server" Text="" Font-Size="Small"></asp:Label>
```

在上面的外观文件中，为 HyperLink 控件和 Label 控件设置了外观，这种设置属于“默认外观”，也就是说这两种控件的外观适用于应用本主题页面上所有的 HyperLink 控件和 Label 控件。

如果想为应用程序中同一类型控件的不同实例应用不同的外观，就要设置“已命名外观”，要在外观文件中为设置的控件指定“SkinID”属性，如下所示：

```
<asp:HyperLink SkinID="hlDiff" runat="server" Font-Size="Small" Font-Underline="false">
</asp:HyperLink>
<asp:Label SkinID="lblDiff" runat="server" Text="" Font-Size="Small"></asp:Label>
```

已命名外观不会自动适用于同类型的所有控件，而是在应用主题的页面控件上通过设置

该控件的 SkinID 属性（如 SkinID="hlDiff"）将已命名外观应用于控件。

2. 应用主题

本项目中所有的网页均应用同一主题，这需要设置 web.config 文件中的 pages 配置节，如下所示：

```
<pages styleSheetTheme="mytheme"></pages>
```

如果只是在单个页面应用主题，则需要在该页页面头部的<% @ Page %>中做如下设置：

```
<% @ Page Language="C#" ... (略) StylesheetTheme="mytheme" %>
```

1.5.10 创建站点地图

在解决方案资源管理器中右击网站 Web，选择“添加新项”，在弹出的对话框的“模板”窗格中选择“站点地图”。

代码清单如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="~/Default.aspx" title="首页" description="">
    <siteMapNode url="~/Search.aspx" title="图书检索" description="" />
    <siteMapNode url="~/ShoppingCart.aspx" title="购物车" description="" />
    <siteMapNode url="~/UserProfiles.aspx" title="修改个人资料" description="" />
    <siteMapNode url="~/MyOrder.aspx" title="我的订单" description="" />
    <siteMapNode url="~/Register.aspx" title="会员注册" description="" />
    <siteMapNode url="~/BookBrief.aspx" title="图书概要" description="">
      <siteMapNode url="~/BookItem.aspx" title="图书详情" description="" />
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

1.5.11 任务小结

通过本节任务的学习，完成了解决方案的构建、网站母版页的设计、整个网站页面与外观的设计和应用等，对.Net Forms 身份验证、成员资格管理、Repeater 控件应用等知识点进行了相关的了解，为深入本次项目开发打下了良好的基础。

1.5.12 练习题

- 1. 使用 ASP.NET 网站配置工具，创建新账户，制作登录 Web 窗体，使用 Login 控件完成登录，并定向到首页。
- 2. 自定义学生信息表，制作 Web 窗体，使用 Repeater 控件完成学生信息的显示。

1.6 前台图书信息浏览、检索实现

学习目标

- 掌握母版页应用、Web 用户控件的定义与应用
- 掌握应用程序调用存储过程
- 掌握动态生成 SQL 语句，完成复合条件查询
- 建议学时：24 学时

1.6.1 任务名称：前台图书信息浏览、检索实现

1.6.2 任务描述

此模块供使用网站的访问者匿名浏览图书概要信息、图书详情信息，或者进行图书检索。前台图书信息浏览、检索实现包括以下几个方面：

- ① 首页对图书信息分 3 个栏目进行显示，这 3 个栏目分别是新书上架、热点推荐、经典书目；
- ② 可以按照栏目或图书类别查看图书概要信息；
- ③ 查看图书详情信息；
- ④ 设置复合条件完成图书检索。

1.6.3 任务分析

① 首页对图书信息分新书上架、热点推荐、经典书目 3 个栏目进行显示，按照图书的添加时间排序，分别显示前 4 位图书。按 3 个栏目显示图书的差别只是栏目条件不同，这也就意味着对不同栏目的前 4 位图书的显示要重复 3 次，是将绝大部分雷同的代码复制粘贴 3 次，还是定义一个用户控件调用 3 次，显然应该选择后者。

② 可以按照栏目或图书类别查看图书概要信息，如前面所述按 3 个栏目显示图书的差别只是栏目条件不同，同理，按图书类别显示图书只是类别编号不同。在完成任务时使用数据源控件 `SqlDataSource`，如果将不同的查询条件定义成参数，使用时传不同的值过去，就会非常灵活地实现按照栏目或图书类别查看图书概要信息，实际上是通过定义一个存储过程来帮助实现该功能。

③ 复合条件查询的关键是按照用户选定的条件动态生成 SQL 语句，我们在公共类类库中编写 `SqlStringConstructor` 类来完成对 SQL 语句的构造。

1.6.4 首页分栏目显示图书信息

1. 效果图

首面效果图如图 1-41 所示。



图 1-41 首页效果图

2. Default.aspx页面中的主要控件及其用途

Default.aspx 页面中的主要控件及其用途如表 1-32 所示。

表 1-32 Default.aspx 页面中的主要控件及其用途

控 件 类 型	控 件 名 称	用 途
DIBook.ascx	DIBook	显示“新书上架”栏目信息
	DIHot	显示“热点推荐”栏目信息
	DIClassic	显示“经典书目”栏目信息
SqlDataSource	SqlldsBook	“新书上架”数据源
	SqlldsHot	“热点推荐”数据源
	SqlldsClassic	“经典书目”数据源

3. 相关知识：使用Web用户控件显示栏目图书信息

如任务分析中所述，我们要把对栏目前 4 位图书的显示通过定义一个用户控件来实现，然后在首页中 3 次调用该控件。

用户控件是能够在其中放置标记和 Web 服务器控件的容器。然后，可以将用户控件作为一个单元对待，为其定义属性和方法。

ASP.NET Web 用户控件与完整的 ASP.NET 网页（.aspx 文件）相似，同时具有用户界面页和代码。可以采取与创建 ASP.NET 页相似的方式创建用户控件，然后向其中添加所需的标记和子控件。用户控件可以像页面一样包含对其内容进行操作（包括执行数据绑定等任务）的代码。

用户控件与 ASP.NET 网页有以下区别：

- ① 用户控件的文件扩展名为 .ascx。
- ② 用户控件中没有 @ Page 指令，而是包含 @ Control 指令，该指令对配置及其他属性进行定义。
- ③ 用户控件不能作为独立文件运行，必须像处理任何控件一样，将它们添加到 ASP.NET 页中。
- ④ 用户控件中没有 html、body 或 form 元素。这些元素必须位于宿主页中。

4. 创建 DIBook.ascx

我们将创建用户控件 DIBook.ascx，并在其中放置绑定数据源的控件 DataList，该 DataList 控件用于显示栏目前 4 位的图书。

在解决方案资源管理器中右击网站 Web，选择“新建文件夹”项，将文件夹命名为 UserControl。

右击文件夹 UserControl，选择“添加新项”菜单项，在弹出对话框的“模板”窗格选择“Web 用户控件”，将控件命名为“DIBook.ascx”，单击“确定”按钮创建该控件。

DIBook.ascx 的 HTML 代码如下：

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="DIBook.ascx.cs" Inherits="User
Ctrl_DIBook" %>
<asp:DataList ID="DataList1" runat="server" RepeatColumns="4"
    DataKeyField="BookId" Width="100%" CellPadding="4" ForeColor="#333333">
    <FooterStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
    <ItemTemplate>
    <table width="100px">
    <tr>
        <td width="100px" height="120px">
            <a href='<%#"BookItem.aspx?BookId="+Eval("BookId")%>'><asp:Image ID="Image1"
runat="server" ImageUrl='<%#"~/Picture/" + Eval("BookImage")%>' Width="90px" Height="120px"/></a>
            </td>
        </tr>
        <tr>
            <td width="100px">
                <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl='<%#"~/BookItem.aspx?
BookId="+Eval("BookId")%>' Font-Size="Small"><%# Eval("BookName")%></asp:HyperLink>
            </td>
        </tr>
    </table>
    </ItemTemplate>
    <AlternatingItemStyle BackColor="White" />
    <ItemStyle BackColor="#FFFBD6" ForeColor="#333333" />
    <SelectedItemStyle BackColor="#FFCC66" Font-Bold="True" ForeColor="Navy" />
    <HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
</asp:DataList>
```

DIBook.ascx.cs 代码清单如下：

```
using System;
```

```
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
public partial class UserControl_DIBook : System.Web.UI.UserControl
{
    private SqlDataSource sqlds;//私有数据成员 SqlDataSource 类对象 sqlds
    public SqlDataSource Sqlds//公有属性，对 sqlds 取值或赋值
    {
        get { return sqlds; }
        set { sqlds = value; }
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        DataList1.DataSource = this.sqlds;//指定 DataList1 的数据源为 sqlds
        DataList1.DataBind();
    }
}
```

5. 在首页中应用母版页及用户控件

① 应用母版页：在网站 Web 中添加新项，模板类型选择“Web 窗体”，命名为“Default.aspx”，勾选“选择母版页”复选框，如图 1-42 所示；在弹出的“选择母版页”对话框中选择“MasterPage.master”，单击“确定”按钮，如图 1-43 所示。

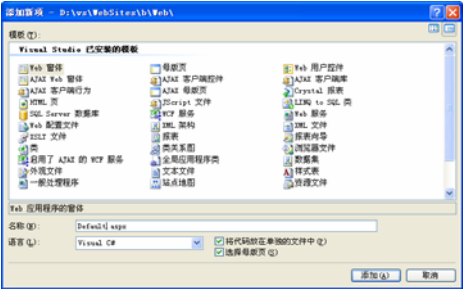


图 1-42 勾选“选择母版页”

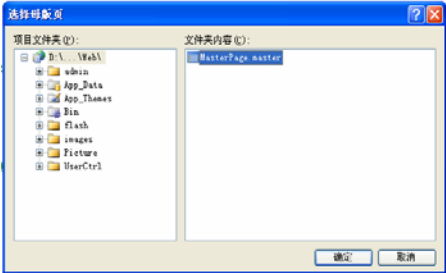


图 1-43 选择母版页

新建的首页 HTML 代码如下：

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true" CodeFile=
"Default.aspx.cs" Inherits="Default" Title="首页" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
```

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

Default 页作为内容页<% @ Page%>中的 MasterPageFile="~/MasterPage.master", 即是应用母版页的属性设置, 母版页会合并到内容页的控件树中。

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
```

</asp:Content>中的内容运行时将会合并到母版页 MasterPage.master 中的<asp:Content PlaceHolder id="head" runat="server"> </asp:ContentPlaceHolder>占位符控件中。

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
```

</asp:Content>中的内容运行时将会合并到母版页 MasterPage.master 中的<asp:Content PlaceHolder id="ContentPlaceHolder1" runat="server"></asp:ContentPlaceHolder>占位符控件中。从用户的角度看, 合并后的母版页和内容页是一个完整的页面。

② 应用 Web 用户控件: 切换到首页的设计视图, 在解决方案资源管理器里将 UserControl 文件中的 DIBook.ascx 控件拖放到首页的相应位置, 在本实例中应拖放 3 次, 用来显示 3 个栏目的前 4 位图书信息。

拖放完成后将在页面自动生成<% @ Register src="UserCtrl/DIBook.ascx" tagname="DIBook" tagprefix="uc1" %>及<uc1:DIBook ID="DIBook" runat="server" />等。如果不用拖放的方法, 也可以直接在源视图中将上面生成的语句添加到 HTML 代码中, 以完成 Web 用户控件的调用。

③ 从工具箱拖放 3 个 SqlDataSource 控件, 按如下代码清单中所示指定属性, 作为数据源备用。

Default.aspx 页完整的 HTML 代码如下:

```
<% @ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true" CodeFile=
"Default.aspx.cs" Inherits="Default2" Title="蓝星网上购书管理系统" %>
<% @ Register src="UserCtrl/DIBook.ascx" tagname="DIBook" tagprefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <div style="text-align:center; font-size:small; margin:15px">
        <p align="left">新书上架: <span style="margin-left:500px;"><a href="BookBrief.aspx?opt=1" target=
        "_blank">更多》》</a></span></p>
        <uc1:DIBook ID="DIBook" runat="server" />
        <asp:SqlDataSource ID="SqlIdsBook" runat="server"
            ConnectionString="<%= $ ConnectionStrings:CVITBOOKSHOPConnectionString2 %>"
            SelectCommand="SELECT top 4 [BookId], [BookName], [BookImage] FROM
[BookBrief] order by AddTime Desc"></asp:SqlDataSource>
        <p align="left">热门推荐: <span style="margin-left:500px;"><a href="BookBrief.aspx?opt=2">更多》》</a></span></p>
        <uc1:DIBook ID="DIHot" runat="server" />
        <asp:SqlDataSource ID="SqlIdsHot" runat="server"
            ConnectionString="<%= $ ConnectionStrings:CVITBOOKSHOPConnectionString2 %>"
            SelectCommand="SELECT top 4 [BookId], [BookName], [BookImage] FROM
[BookBrief] where IsHeadLine='true' order by AddTime Desc "></asp:SqlDataSource>
```



```

<p align="left">经典书目: <span style=" margin-left:500px;"><a href="BookBrief.aspx?opt=3">更多》》
</a></span> </p>

<uc1:DIBook ID="DIClassic" runat="server" />
<asp:SqlDataSource ID="SqlDsClassic" runat="server"
    ConnectionString="<%$ ConnectionStrings:CVITBOOKSHOPConnectionString2 %>"
    SelectCommand="SELECT top 4 [BookId], [BookName], [BookImage] FROM
[BookBrief] where IsClassic='true' order by AddTime Desc"></asp:SqlDataSource>

</div>
</asp:Content>

```

6. Default.aspx.cs代码实现

```

using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            DIBook.Sqlds = SqldsBook;//绑定数据源为新书上架栏目信息
            DIHot.Sqlds = SqldsHot;//绑定数据源为热门推荐栏目信息
            DIClassic.Sqlds = SqldsClassic;// 绑定数据源为经典书目信息

            //Default.aspx 是默认的登录页，当匿名用户访问需要身份验证的网页时，会自动跳转到该页登录，
            使用以下代码片断显示要求登录的提示信息
            if (Request.Url.AbsoluteUri.ToString().Contains("ReturnUrl") && Request.Url.Absolute
Uri.ToString().Contains("admin"))
                this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert('您尚
未以管理员身份登录，请登录后进行此操作!')</script>");
            else if (Request.Url.AbsoluteUri.ToString().Contains("ReturnUrl"))
            {
                this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert('您
尚未以会员身份登录，请登录后进行此操作!')</script>");
            }
        }
    }
}

```



```

)
AS
if @CategoryId=-1 and @opt=-1
select * from BookBrief
else if
@CategoryId=-1 and @opt=1
select top 100 * from BookBrief order by AddTime desc
else if
@CategoryId=-1 and @opt=2
select * from BookBrief where IsHeadLine='true' order by AddTime desc
else if
@CategoryId=-1 and @opt=3
select * from BookBrief where IsClassic='true' order by AddTime desc
else
select * from BookBrief where CategoryId=@CategoryId

```

@CategoryId 表示图书类别编号，@opt 为栏目编号（1——新书上架；2——热点推荐；3——经典书目），依据传来的参数值的不同执行不同的 SQL 语句。二者均为-1 时，显示所有的图书概要信息；@CategoryId=-1 且@opt=1 时，选择最新的图书信息前 100 条；@CategoryId=-1 且@opt=2 时，选择热点推荐的图书；@CategoryId=-1 且@opt=3 时，选择经典书目；其余情况按图书类别编号选择该类别图书信息。

② 进行 SqlDataSource 控件设置：当在首页上单击栏目“更多”的链接时，会向 BookBrief.aspx 页面传递参数 opt，同理当单击图书某类别的链接时，会向 BookBrief.aspx 页面传递参数 CategoryId。因此，设置 SqlDataSource 控件时要做如下选择，如图 1-45 和图 1-46 所示。

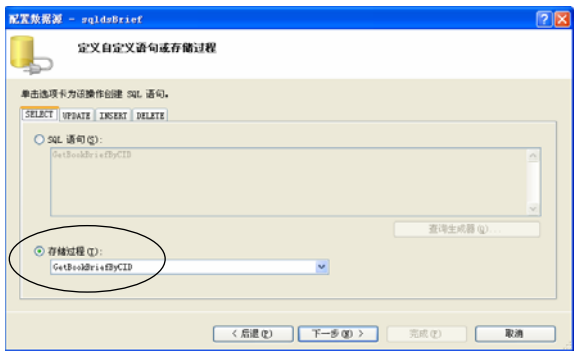


图 1-45 选择“存储过程”

设置之后会在源视图生成如下代码：

```

<asp:SqlDataSource ID="sqlldsBrief" runat="server"
    ConnectionString="<%$ ConnectionStrings:CVITBOOKSHOPConnectionString %>"
    SelectCommand="GetBookBriefByCID" SelectCommandType="StoredProcedure">
    <SelectParameters>
        <asp:QueryStringParameter DefaultValue="-1" Name="CategoryId"
            QueryStringField="CategoryId" Type="Int16" />
        <asp:QueryStringParameter DefaultValue="-1" Name="opt"

```

```

QueryStringField="opt" Type="Int16" />
</SelectParameters>
</asp.SqlDataSource>

```

将用来显示图书概要信息的 GridView 控件 gvBrief 的 DataSourceID 属性设置为“sqldsBrief”，完成绑定数据源。

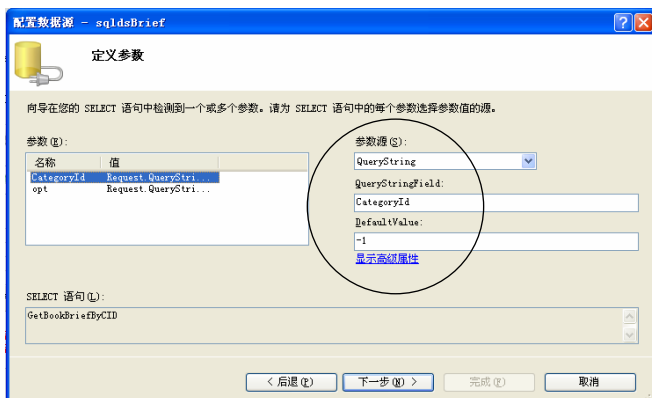


图 1-46 设置“参数源”

4. BookBrief.aspx完整的HTML代码

```

<% @ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true" CodeFile=
"BookBrief.aspx.cs" Inherits="BookBrief" Title="图书概要信息" Theme %>
<% @ Register src="UserCtrl/PS.ascx" tagname="PS" tagprefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <uc1:PS ID="Location" runat="server" />
    <div style="width:auto; text-align:center; padding-top:10px;">
        <asp:Label ID="lblBrief" runat="server" Text="" Font-Size="Small"></asp:Label>
        <asp:GridView ID="gvBrief" runat="server" AllowPaging="True"
            AutoGenerateColumns="False" CellPadding="4" DataSourceID="sqldsBrief"
            ForeColor="#333333" GridLines="None" PageSize="4" Width="95%"
            Font-Size="Small">
            <FooterStyle BackColor="#506CD1" Font-Bold="True" ForeColor="White" />
            <RowStyle BackColor="#EFF3FB" />
            <Columns>
                <asp:TemplateField HeaderText="图片" SortExpression="图片">
                    <EditItemTemplate>
                        <asp:TextBox ID="TextBox1" runat="server" Text=<%# Bind("BookImage")
                    %>></asp:TextBox>
                    </EditItemTemplate>
                    <ItemTemplate>
                        <a href='<%#"BookItem.aspx?BookId="+Eval("BookId")%>'><asp:Image ID=
"Image1" runat="server" ImageUrl='<%#"~/Picture/" + Eval("BookImage")%>' Width="70px" Height="100px"
                    /></a>

```

```

        </ItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="书名" SortExpression="BookName">
        <ItemTemplate>
            <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl= '<%#~/BookItem.
aspx?BookId="+Eval("BookId")%>' Font-Size="Small"><asp:Label ID="Label1" runat="server" Text='<%# Bind
("BookName") %>'></asp:Label></asp:HyperLink>
        </ItemTemplate>
    </EditItemTemplate>
        <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl= '<%#~/BookItem.
aspx?BookId="+Eval("BookId")%>' Font-Size="Small"><asp:TextBox ID="TextBox2" runat="server" Text='<%#
Bind("BookName") %>'></asp:TextBox></asp:HyperLink>
    </EditItemTemplate>
</asp:TemplateField>
<asp:BoundField DataField="BookDesc" HeaderText="描述" SortExpression="BookDesc" />
<asp:BoundField DataField="AddTime" HeaderText="添加时间" SortExpression=" Add
Time" />
<asp:CheckBoxField DataField="IsHeadLine" HeaderText="热点" SortExpression="IsHead
Line" />
<asp:CheckBoxField DataField="IsClassic" HeaderText="经典" SortExpression="IsClassic"/>
</Columns>
<PagerStyle BackColor="#2461BF" ForeColor="White"
    HorizontalAlign="Center" />
<SelectedRowStyle BackColor="#D1DDF1" ForeColor="#333333" Font-Bold="True" />
<HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
<EditRowStyle BackColor="#2461BF" />
<AlternatingRowStyle BackColor="PaleGoldenrod" />
</asp:GridView>
<asp:SqlDataSource ID="sqldsBrief" runat="server"
    ConnectionString="<%= $ ConnectionStrings:CVITBOOKSHOPConnectionString %>"
    SelectCommand="GetBookBriefByCID" SelectCommandType="StoredProcedure">
    <SelectParameters>
        <asp:QueryStringParameter DefaultValue="-1" Name="CategoryId"
            QueryStringField="CategoryId" Type="Int16" />
        <asp:QueryStringParameter DefaultValue="-1" Name="opt"
            QueryStringField="opt" Type="Int16" />
    </SelectParameters>
</asp:SqlDataSource>
</div>
</asp:Content>

```

5. BookBrief.aspx.cs代码实现

```

using System;
using System.Collections;
using System.Configuration;
using System.Data;

```

```

using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class BookBrief : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            lblBrief.Text="当前共有"+((DataView)this.sqldsBrief.Select(DataSourceSelect Arguments.
Empty)).Count.ToString() + "条记录";
        }
    }
}

```

1.6.6 图书详情信息查看

在图书概要页上单击 `gvBrief` 数据行中书的图片或者书名，都可以打开图书详情页 `BookItem.aspx`，并向该页传递表示书号的参数 `BookId`，其值为数据行中所对应的书号字段的值。在首页单击书的图片同样可以打开相应的图书详情页面。创建新的 Web 窗体及应用母版页同前所述。

1. 图书详情页BookItem.aspx效果图

图书详情页 `BookItem.aspx` 的效果图如图 1-47 所示。



图 1-47 图书详情页 BookItem.aspx 效果图

2. BookItem.aspx页面中各个控件的属性设置及其用途

BookItem.aspx 页面中各个控件的属性设置及其用途如表 1-34 所示。

3. 相关知识：使用DetailsView 控件显示数据源的单个记录

用 DetailsView 控件显示数据源的单个记录，其中每个数据行表示记录中的一个字段。使用 DetailsView 控件，可以从它的关联数据源中一次显示、编辑、插入或删除一条记录。默认情况下，DetailsView 控件将记录的每个字段显示在它自己的一行内。

表 1-34 BookItem.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
PS.ascx	Location		站点导航
DetailsView	dvItem	DataSourceID="sqldsBrief"	显示图书概要信息
SqlDataSource	sqldsItem	SelectCommandType="StoredProcedure"执行存储过程	数据源

DetailsView 控件提供以下用于绑定到数据的选项。

- 使用 DataSourceID 属性进行数据绑定，此选项能够将 DetailsView 控件绑定到数据源控件。建议使用此选项，因为它允许 DetailsView 控件利用数据源控件的功能并提供了内置的更新和分页功能。
- 使用 DataSource 属性进行数据绑定，此选项能够绑定到包括 ADO.NET 数据集和数据读取器在内的各种对象。此方法需要用户为任何附加功能（如更新和分页等）编写代码。

当使用 DataSourceID 属性绑定到数据源时，DetailsView 控件支持双向数据绑定。除可以使该控件显示数据之外，还可以使它自动支持对绑定数据的插入、更新和删除操作。

- ① 从工具箱拖放 SqlDataSource 到 BookBrief.aspx 页面，设置 SQL 语句、参数及参数源，如图 1-48 和图 1-49 所示。
- ② 从工具箱拖放 DetailsView 控件到 BookBrief.aspx 页面，设置数据源并编辑字段，如图 1-50 和图 1-51 所示。



图 1-48 自定义 SQL 语句



图 1-49 定义参数源

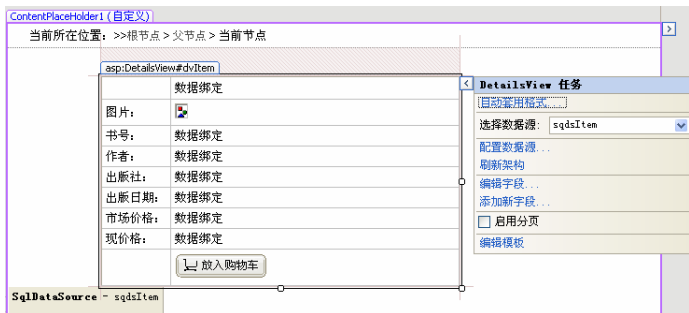


图 1-50 DetailsView 控件

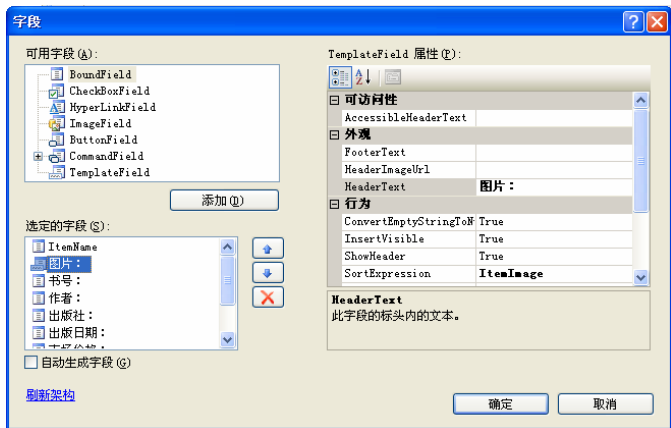


图 1-51 编辑 DetailsView 控件字段

4. BookItem.aspx页面的HTML代码

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true" CodeFile=
"BookItem.aspx.cs" Inherits="BookItem" Title="图书详情信息" %>
<%@ Register src="UserCtrl/PS.ascx" tagname="PS" tagprefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<uc1:PS ID="PS1" runat="server" />
```



```

<div style="font-size:small; margin-top:30px; margin-left:100px;">
  <asp:DetailsView ID="dvItem" runat="server" Height="50px"
    DataSourceID="sqdsItem" AutoGenerateRows="False" BackColor="White"
    BorderColor="#CCCCCC" BorderStyle="None" BorderWidth="1px" CellPadding="4"
    ForeColor="Black" GridLines="Horizontal">
    <FooterStyle BackColor="#CCCC99" ForeColor="Black" />
    <PagerStyle BackColor="White" ForeColor="Black" HorizontalAlign="Right" />
    <Fields>
      <asp:BoundField DataField="ItemName" SortExpression="ItemName"
        ItemStyle-Width="300px" >
<ItemStyle Width="300px"></ItemStyle>
      </asp:BoundField>
      <asp:TemplateField HeaderText="图片： " SortExpression="ItemImage">
        <ItemTemplate>
          <asp:Image ID="Image1" runat="server" ImageUrl='<%# "~/picture/" + Eval
("ItemImage")%>' />
        </ItemTemplate>
      </asp:TemplateField>
      <asp:BoundField DataField="BookId" HeaderText="书号： " SortExpression="BookId" />
      <asp:BoundField DataField="BookAuthor" HeaderText="作者： "
        SortExpression="BookAuthor" />
      <asp:BoundField DataField="Publisher" HeaderText="出版社： "
        SortExpression="Publisher" />
      <asp:BoundField DataField="PublishDate" HeaderText="出版日期： "
        SortExpression="PublishDate" />
      <asp:BoundField DataField="BookPrice" HeaderText="市场价格： "
        SortExpression="BookPrice" />
      <asp:BoundField DataField="ListPrice" HeaderText="现价格： "
        SortExpression="ListPrice" />
      <asp:TemplateField>
        <ItemTemplate>
          <asp:HyperLink ID="hlCart" runat="server" NavigateUrl='<%# "~/ShoppingCart.aspx?
BookId="+Eval("BookId")%>'></asp:HyperLink>
        </ItemTemplate>
      </asp:TemplateField>
    </Fields>
    <HeaderStyle BackColor="#333333" Font-Bold="True" ForeColor="White" />
    <EditRowStyle BackColor="#CC3333" Font-Bold="True" ForeColor="White" />
  </asp:DetailsView>
</div>
<asp:SqlDataSource ID="sqdsItem" runat="server"
  ConnectionString="<%= $ ConnectionStrings:CVITBOOKSHOPConnectionString %>"
  SelectCommand="SELECT [BookId], [BookAuthor], [Publisher], [PublishDate], [BookPrice],
[ListPrice], [ItemName], [ItemImage] FROM [Item] WHERE ([BookId] = @BookId)">
  <SelectParameters>
    <asp:QueryStringParameter Name="BookId" QueryStringField="BookId"
      Type="String" />

```

```
</SelectParameters>
</asp:SqlDataSource>
</asp:Content>
```

5. BookItem.aspx.cs代码实现

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class BookItem : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //为转到购物车页面防刷新、设置 Session 对象初始值
        Session["flag"] = "";
    }
}
```

1.6.7 复合条件图书检索

在解决方案资源管理器网站 Web 中创建图书检索页 Search.aspx，勾选应用母版页。

1. 图书检索页Search.aspx效果图

图书检索页 Search.aspx 的效果图如图 1-52 所示。



图 1-52 图书检索页 Search.aspx 的效果图

2. 在Common类库中创建公共类SqlStringConstructor类构造SQL语句

在 Common 类库中新建 SqlStringConstructor 类，代码清单如下：

```
using System;
using System.Collections;

namespace BookShop.Common
{
    /// <summary>
    /// SQLString 的摘要说明
    /// </summary>
    public class SqlStringConstructor
    {
        /// <summary>
        /// 公有静态方法，将文本转换成适合在 SQL 语句里使用的字符串
        /// </summary>
        /// <returns>转换后文本</returns>
        public static String GetQuotedString(String pStr)
        {
            return ("'" + pStr.Replace("'", "''") + "'");
        }

        /// <summary>
        /// 根据条件哈希表，构造 SQL 语句中的 AND 条件子句
        /// </summary>
        /// <param name="conditionHash">条件哈希表</param>
        /// <returns>AND 关系条件子句</returns>
        public static String GetConditionClause(Hashtable queryItems)
        {
            int Count = 0;
            String Where = "";
            //根据哈希表，循环生成条件子句
            foreach (DictionaryEntry item in queryItems)
            {
                if (Count == 0)
                    Where = " where ";
                else
                    Where += " and ";

                //根据查询列的数据类型，决定是否加单引号
                if (item.Value.GetType().ToString() == "System.String")
                {
                    Where += item.Key.ToString()
                        + " like "
                        + SqlStringConstructor.GetQuotedString("%"
                        + item.Value.ToString()
```

```

        + "%");
    }
    else if (item.Value.GetType().ToString() == "System.DateTime[]")
    {
        string[] time = item.Value.ToString().Split(',');
        Where += item.Key.ToString()
            + " between "
            +
        SqlStringConstructor.GetQuotedString(((DateTime[])item.Value)[0].ToString())+"and"+
        SqlStringConstructor.GetQuotedString(((DateTime[])item.Value)[1].ToString());
    }
    else
    {
        Where += item.Key.ToString() + " = " + item.Value.ToString();
    }
    Count++;
}
return Where;
}
}
}

```

在解决方案资源管理器中右击“Common”类库，选择“重新生成”，然后在网站 Web 中添加引用，具体操作如图 1-53 和图 1-54 所示。

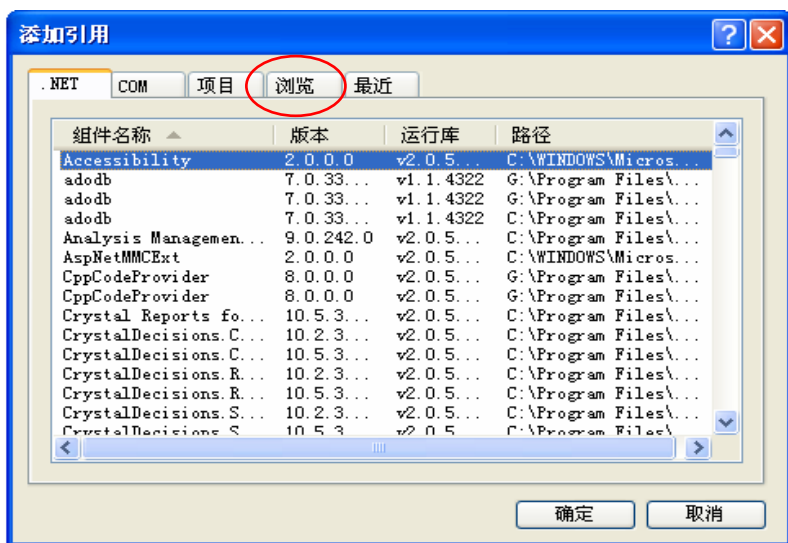


图 1-53 选取位置添加引用

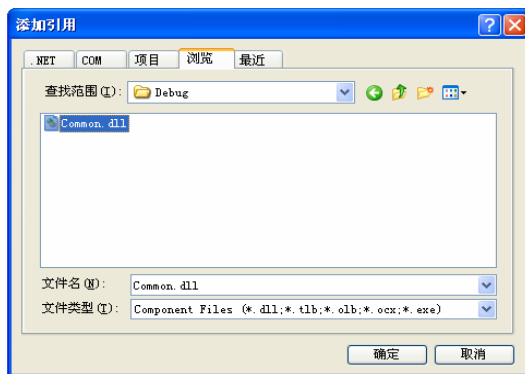


图 1-54 选择 dll 文件

3. 在 DAL 类库中创建图书详情数据访问类 ItemAccess 类

在 DAL 类库中创建图书详情数据访问类 ItemAccess 类，并定义 Search 方法：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BookShop.Model;
using System.Data.SqlClient;
using System.Data;
using BookShop.Common;
namespace BookShop.DAL
{
    public class ItemAccess
    {
        /// <summary>
        /// 图书检索方法
        /// </summary>
        /// <param name="sql">SQL 语句</param>
        /// <returns>DataSet</returns>

        public DataSet Search(string sql)
        {
            DataSet ds = new DataSet();//创建数据集对象
            SqlDataAdapter da = null;//创建数据适配器对象
            DataBase db = new DataBase();//创建数据库连接类实例
            da = db.CreateDataAdapter(CommandType.Text, sql, null, 1);//获得用于查询的数据适
            配器对象
            da.Fill(ds);//将查询结果填充到数据集中
            return ds; //返回数据集
        }
    }
}
```

编写代码后，在解决方案资源管理器中右击 DAL，选择“生成”菜单项。

4. 在BLL类库中创建图书详情业务逻辑类ItemManager类

添加对 DAL 类库的引用，在 BLL 类库中创建图书详情业务逻辑类 ItemManager 类：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BookShop.DAL;
using BookShop.Model;
using System.Data;

namespace BookShop.BLL
{
    [Serializable]
    public class ItemManager
    {
        //私有成员图书详情数据访问类对象
        private static readonly ItemAccess dal = new ItemAccess();
        //图书检索方法
        public DataSet Search(string sql)
        {
            return dal.Search(sql);//调用数据访问类的图书检索方法
        }
    }
}
```

编写代码后，在解决方案资源管理器中右击 BLL，选择“生成”菜单项。

5. 在网站Web中创建图书检索页Search.aspx

Search.aspx 页面中各个控件的属性设置及其用途如表 1-35 所示。

表 1-35 Search.aspx 页面中各个控件的属性设置及其用途

控件类型	控件名称	主要属性设置	用途
PS.ascx	Location		站点导航
CheckBox	cbTitle	Text 属性设置为“按书名查找记录”	设置查询关键字
	cbAuthor	Text 属性设置为“按作者查找记录”	设置查询关键字
	cbPress	Text 属性设置为“按出版社查找记录”	设置查询关键字
TextBox	txtTitle	Text 属性设置为“ ”	设置关键字的值
	txtAuthor	Text 属性设置为“ ”	设置关键字的值
	txtPress	Text 属性设置为“ ”	设置关键字的值
Button	btnOk	Text 属性设置为“确定”	执行查询
	btnReset	Text 属性设置为“重置条件”	重置条件
GridView	gvSearch	AllowPaging="True" PageSize="4"	显示符合条件的图书信息
Label	lblShow	Text 属性设置为“ ”	显示符合条件的记录条数

Search.aspx 页面的 HTML 代码如下：

```

<% @PageLanguage="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true" CodeFile=
"Search.aspx.cs" Inherits="Search" Title="图书检索" %>
<% @ Register src="UserCtrl/PS.ascx" tagname="PS" tagprefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:ContentID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <uc1:PS ID="Location" runat="server" />
    <asp:Panel ID="Panel1" runat="server" Width="100%">
        <table id="Table1" style="WIDTH:90%;margin-left:25px; margin-top:20px;" border="0">
<tr>
    <td width="150px;">
<asp:checkboxid="cbTitle"runat="server"Font-Size="Small"Text="按书名查找记录"
"oncheckedchanged="cbTitle_CheckedChanged"></asp:checkbox></td>
<td align="right">
    <asp:label id="lblTitle" runat="server" Font-Size="Small">书名</asp:label></td>
<td > <asp:textbox id="txtTitle" runat="server"></asp:textbox>    </td>
</tr>
<tr><td>
<asp:checkboxid="cbAuthor"runat="server"Font-Size="Small"Text="按作者查找记录"
"oncheckedchanged="cbAuthor_CheckedChanged" ></asp:checkbox></td>
<td align="right">
<asp:LabelID="lblAuthor"runat="server" Text="作者" Font-Size="Small"></asp:Label></td>
<td>
    <asp:TextBox ID="txtAuthor" runat="server"></asp:TextBox></td>
</tr>
<tr>
<td ><asp:checkbox id="cbPress" runat="server" Font-Size="Small" Text="按出版社查找记录"
OnCheckedChanged="cbPress_CheckedChanged"></asp:checkbox></td>
    <td align="right" width="75px"><asp:Label ID="lblPress" runat="server" Text="出版社"
Font-Size="Small"></asp:Label></td>
    <td >
        <asp:TextBox ID="txtPress" runat="server"></asp:TextBox>
    </td>
</tr>
</table>
<table style="WIDTH:90%; HEIGHT: auto; margin-left:25px">
<tr><td style="width: 564px"><br />
<asp:buttonid="btnOk"runat="server"Text="确定"OnClick="btnOk_Click"Width="80px"
CausesValidation="false"></asp:button>
<asp:buttonid="btnReset"runat="server"Text="重置条件"onclick="btnReset_Click"
CausesValidation="false"></asp:button>&nbsp;
    </td></tr>
</table>
</asp:Panel>
<div style="margin-left:25px;">
    <asp:Label ID="lblShow" runat="server" Text="" Font-Size="small"></asp:Label>
    <br />

```

```

<asp:GridView ID="gvSearch" runat="server" AllowPaging="True"
AutoGenerateColumns="False" CellPadding="4"
ForeColor="#333333" GridLines="None" PageSize="4" Width="95%"
    Font-Size="Small" onpageindexchanging="gvSearch_PageIndexChanging">
<FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
<RowStyle BackColor="#EFF3FB" />
<Columns>
    <asp:TemplateField HeaderText="图片" SortExpression="图片">
        <EditItemTemplate>
            <asp:TextBox ID="TextBox1" runat="server" Text='<%# Bind("ItemImage")
%>'></asp:TextBox>
        </EditItemTemplate>
    </ItemTemplate>
    <a href='<%#"BookItem.aspx?BookId="+Eval("BookId")%>'><asp:Image ID="Image1"
runat="server" ImageUrl='<%#"~/Picture/"+Eval("ItemImage")%>' Width="70px" Height="100px" /
></a>
    </ItemTemplate>
    <asp:TemplateField>
        <asp:TemplateField HeaderText="书名" SortExpression="ItemName">
            <ItemTemplate>
                <asp:HyperLinkID="HyperLink1"runat="server"NavigateUrl='<%#"~/BookItem.aspx?BookId="+Eval
("BookId")%>' Font-Size="Small"><asp:Label ID="Label1" runat="server" Text='<%#
Bind("ItemName") %>'></asp:Label></asp:HyperLink>
            </ItemTemplate>
            <EditItemTemplate>
                <asp:HyperLinkID="HyperLink1"runat="server"NavigateUrl='<%#"~/BookItem.aspx?BookId="+Eval
("BookId")%>'
                Font-Size="Small"><asp:TextBoxID="TextBox2"runat="server"Text='<%#Bind("ItemName")%>'></a
sp:TextBox></asp:HyperLink>
            </EditItemTemplate>
        </asp:TemplateField>
        <asp:BoundField DataField="BookAuthor" HeaderText="作者"
SortExpression="BookAuthor" />
        <asp:BoundField DataField="Publisher" HeaderText="出版社"
SortExpression="Publisher" />
        <asp:BoundField DataField="PublishDate" HeaderText="出版时间"
SortExpression="PublishDate" />
        <asp:BoundField DataField="ListPrice" HeaderText="售价"
SortExpression="ListPrice" />
    </Columns>
    <PagerStyle BackColor="#2461BF" ForeColor="White"
        HorizontalAlign="Center" />
    <SelectedRowStyle BackColor="#D1DDF1" ForeColor="#333333" Font-Bold="True" />
    <HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
    <EditRowStyle BackColor="#2461BF" />
    <AlternatingRowStyle BackColor="PaleGoldenrod" />
</asp:GridView>

```



```
</div>
</asp:Content>
```

Search.aspx.cs 代码如下：

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Drawing;
using BookShop.Common;
using BookShop.BLL;

public partial class Search : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            string gsql = ""; //该变量的值为构造的 SQL 语句
            if (ViewState["gsql"] != null)
            {
                gsql = ViewState["gsql"].ToString();
                BindGrid(gsql);
            }
        }
        //使用 GridView 控件绑定数据源
        //参数 gsql 为构造好的 SQL 语句
        private void BindGrid(string gsql)
        {
            ItemManager im = new ItemManager(); //创建图书详情业务逻辑类对象
            DataSet ds = new DataSet(); //创建数据集对象
            ds = im.Search(gsql); //调用业务逻辑类的图书检索方法，返回查询结果
            gvSearch.DataSource = ds;
            gvSearch.DataBind();
            lblShow.Text = "共找到" + ds.Tables[0].Rows.Count.ToString() + "条符合条件的记录";
            lblShow.ForeColor = Color.Blue;
        }
    }
}
```

//单击“确定”按钮执行查询

```
protected void btnOk_Click(object sender, EventArgs e)
{
    gvSearch.Visible = true;
    string gsql = "select * from Item";
    Hashtable ht = new Hashtable(); //创建哈希表对象，保存查询条件的关键字与值
    if (cbTitle.Checked) //如果按书名查询被勾选
        ht.Add("ItemName", txtTitle.Text.Trim());
    if (cbAuthor.Checked) //如果按作者查询被勾选
        ht.Add("BookAuthor", txtAuthor.Text.Trim());
    if (cbPress.Checked) //如果按出版社查询被勾选
    {
        ht.Add("Publisher", txtPress.Text.Trim());
    }
    //构造 SQL 语句
    gsql = gsql + SqlStringConstructor.GetConditionClause(ht)+" order by PublishDate Desc";
    ViewState["gsql"] = gsql;
    BindGrid(gsql);
}
```

//单击“重置条件”按钮，将查询条件设置为默认状态

```
protected void btnReset_Click(object sender, EventArgs e)
{
    cbTitle.Checked = false;
    cbTitle_CheckedChanged(cbTitle, e);
    cbAuthor.Checked = false;
    cbAuthor_CheckedChanged(cbAuthor, e);
    cbPress.Checked = false;
    cbPress_CheckedChanged(cbPress, e);
    gvSearch.Visible = false;
    lblShow.Text = "";
}

protected void cbTitle_CheckedChanged(object sender, EventArgs e)
```

```
{
    //如果按书名查询条件被取消，将值域清空
    if (cbTitle.Checked == false) txtTitle.Text = "";
}
```

```
protected void cbPress_CheckedChanged(object sender, EventArgs e)
```

```
{
    //如果按出版社查询条件被取消，将值域清空
    if (cbPress.Checked == false) txtPress.Text = "";
}
```

```
protected void cbAuthor_CheckedChanged(object sender, EventArgs e)
```

```
{
    //如果按作者查询条件被取消，将值域清空
    if (cbAuthor.Checked == false) txtAuthor.Text = "";
}
```

```
protected void gvSearch_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    gvSearch.PageIndex = e.NewPageIndex;
    string gsql = ViewState["gsql"].ToString();
    BindGrid(gsql);
}
}
```

1.6.8 任务小结

通过本任务的学习完成了前台图书信息的浏览，其中主要包括首页分栏目显示图书信息、图书概要信息查看、图书详情信息查看及图书复合条件检索。关于数据绑定控件和数据源控件的使用、参数源的设置，以及对存储过程的调用、动态生成 SQL 语句等是本章的重要知识点。

1.6.9 练习题

1. 独立完成快速搜索功能的实现并制作成 Web 用户控件。
2. 自定义学生成绩信息表，应用 GridView 控件完成分页显示学生成绩信息。

1.7 前台用户的注册、修改个人资料实现

学习目标

- 掌握 CreateUserWizard 控件的应用
- 掌握 .Net 角色权限管理
- 掌握 Membership 类的应用
- 建议学时：8 学时

1.7.1 任务名称：前台用户的注册、修改个人资料实现

1.7.2 任务描述

创建或管理角色，管理角色的访问权限，注册为会员并能够修改个人资料。

1.7.3 任务分析

本应用程序的使用者角色为后台管理员和会员，首先要在 ASP.net 配置工具中创建角色，并在 Web.config 文件中进行配置，存储到自定义的 SQL 数据库中。不同的角色有不同的访问权限，会员不能访问后台管理页面，要在 ASP.net 配置工具中进行角色权限设置。注册的会员可以修改个人资料，应用 MembershipUser 类和 Membership 类实现。

1.7.4 创建或管理角色及设置角色的访问权限

① 打开“ASP.net 网站管理工具”，选择“安全”选项卡，如图 1-55 所示，选择“创建或管理角色”链接。

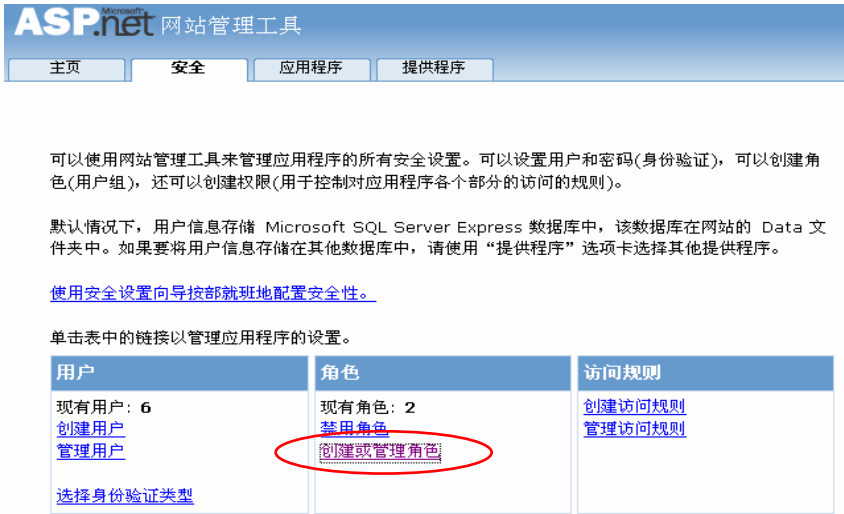


图 1-55 使用“ASP.net 网站管理工具”创建或管理角色

② 创建管理员角色“admin”与会员角色“normal”。



图 1-56 创建新角色

③ 管理角色权限。在“安全”选项卡下选择“创建访问规则”链接，如图 1-57 所示。

可以使用网站管理工具来管理应用程序的所有安全设置。可以设置用户和密码(身份验证)，可以创建角色(用户组)，还可以创建权限(用于控制对应用程序各个部分的访问的规则)。

默认情况下，用户信息存储 Microsoft SQL Server Express 数据库中，该数据库在网站的 Data 文件夹中。如果要将在用户信息存储在其他数据库中，请使用“提供程序”选项卡选择其他提供程序。

使用安全设置向导按部就班地配置安全性。

单击表中的链接以管理应用程序的设置。

用户	角色	访问规则
现有用户: 6 创建用户 管理用户 选择身份验证类型	现有角色: 2 禁用角色 创建或管理角色	创建访问规则 管理访问规则

图 1-57 设置访问规则

④ 添加访问规则。选择“Web”根目录，在“角色”下拉框中选择“admin”，权限设置为“允许”，如图 1-58 所示。



图 1-58 添加访问规则

⑤ 同上，添加访问规则。选择“admin”目录（包含后台管理页面），分别为会员角色 normal 和匿名用户设置“拒绝”权限，如图 1-59 所示。



图 1-59 访问规则列表

1.7.5 会员注册

在网站 Web 中新建 Web 窗体，命名为 Register.aspx，从工具箱中拖放 CreateUserWizard 控件到该页面，并展开设置面板自动套用格式“专业型”，如图 1-60 所示。



图 1-60 会员注册页 Register.aspx

CreateUserWizard 控件封装了创建用户的功能，不用编码便可实现成员资格用户的创建，用户信息存储到默认的 SQL 数据库中。因为本项目启用了角色，而创建的用户要被加入到普通会员 normal 中，所以要通过编码实现这一功能。

1. Register.aspx.cs代码实现

```
public partial class Register : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void CreateUserWizard1_ContinueButtonClick(object sender, EventArgs e)
    {
        CreateUserWizard1.ActiveStepIndex = 0;
        CreateUserWizard1.UserName = "";
        CreateUserWizard1.Email = "";
        CreateUserWizard1.Question = "";
    }

    protected void CreateUserWizard1_CreatingUser(object sender, LoginCancelEventArgs e)
    {
        try
        {
            //将创建的用户加入到普通会员 normal 角色中
            Roles.AddUserToRole(CreateUserWizard1.UserName,"normal");
        }
        catch (Exception ex)
        {
            //显示错误信息
            lblMeg.Text = ex.Message;
            lblMeg.ForeColor = Color.Red;
            lblMeg.Font.Size = 8;
        }
    }
}
```

```

//取消注册
e.Cancel = true;
    }
}
}

```

2. Web.config配置

在 Web.config 文件中，同 membership 配置节一样对角色管理 roleManager 做如下配置，将把用户角色信息保存到用户自定义的 SQL 数据库中。

```

<roleManager defaultProvider="SqlProvider"
    enabled="true"
    cacheRolesInCookie="true"
    cookieName=".ASPROLES"
    cookieTimeout="20"
    cookiePath="/"
    cookieRequireSSL="false"
    cookieSlidingExpiration="true"
    cookieProtection="All">
    <providers>
    <add name="SqlProvider"
        type="System.Web.Security.SqlRoleProvider"
        connectionStringName=" CVITBOOKSHOPConnectionString "
        applicationName="/" />
    </providers>
</roleManager>

```

1.7.6 修改会员个人资料

新建 Web 窗体 UserProfiles.aspx，完成修改会员个人资料的功能。

1. 修改个人资料页UserProfiles.aspx效果图

修改个人资料页 UserProfiles.aspx 的效果图如图 1-61 所示。



图 1-61 修改个人资料页 UserProfiles.aspx 的效果图

2. UserProfiles.aspx页面中各个控件的属性设置及其用途

UserProfiles.aspx 页面中如表 1-36 所示。

表 1-36 UserProfiles.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
PS.ascx	Location		站点导航
TextBox	txtFPwd	Text 属性设置为 “ ”	输入原密码
	txtNPwd	Text 属性设置为 “ ”	输入新密码
	txtCPwd	Text 属性设置为 “ ”	确认新密码
Button	btnUpdate	Text 属性设置为 “修改”	执行修改
	btnReset	Text 属性设置为 “重置”	重置
Label	lblTitle	Text 属性设置为 “ ”	显示说明信息
CompareValidator	cvPwd	ErrorMessage 属性设置为 “二次密码输入不一致” ControlToCompare 属性值 “txtNPwd” ControlToValidate 属性值 “txtCPwd”	比较验证
ValidationSummary	ValidSummary	ShowMessageBox="True" ShowSummary="False"	显示错误信息提要

3. UserProfiles.aspx. cs代码实现

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Drawing;

public partial class SheZhi : System.Web.UI.Page
{
    string name = "";
    MembershipUser mu = null;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            //将母版页验证控件关闭
            RequiredFieldValidator rv = (RequiredFieldValidator)(this.Master.FindControl("rfvName"));
            RequiredFieldValidator rv2 = (RequiredFieldValidator)(this.Master.FindControl("rfvPwd"));
            rv.Enabled = false;
            rv2.Enabled = false;
        }
    }
}
```



```

//获取当前通过身份验证的用户标识
name = HttpContext.Current.User.Identity.Name;
ViewState["name"] = name;
lblTitle.Text = "修改用户[" + name + "]的个人信息";
lblTitle.Font.Size = 10;
lblTitle.ForeColor = Color.Red;
lblTitle.Font.Name = "新宋体";
//获取用户标识为当前通过验证的用户标识的 MembershipUser 对象
mu = Membership.GetUser(name);
ViewState["mu"] = mu;
txtEmail.Text = mu.Email;
}
Response.CacheControl = "no-cache";
}

//修改个人资料
protected void btnUpdate_Click(object sender, EventArgs e)
{
    name = ViewState["name"];
    mu = ViewState["mu"];
    try
    {
        //验证原密码正确
        if (Membership.ValidateUser(name,txtFPwd.Text.Trim()))
        {
            //更改用户密码
            mu.ChangePassword(txtFPwd.Text.Trim(),txtNPwd.Text.Trim());
            //设置电邮地址为新值
            mu.Email = txtEmail.Text.Trim();
            //更新用户个人资料
            Membership.UpdateUser(mu);
            this.ClientScript.RegisterStartupScript(this.GetType(), "", "<Script>window.alert('修改成功')</Script>");
        }
        else
        {
            this.ClientScript.RegisterStartupScript(this.GetType(), "", "<Script>window.alert('您输入的原密码错误!请重新输入')</Script>");
        }
    }
    catch (Exception d)
    {
        Response.Write(d.Message);
    }
}

//重置
protected void btnReset_Click(object sender, EventArgs e)

```

```
{  
    txtEmail.Text = "";  
}  
}
```

1.7.7 任务小结

通过本任务的学习，完成应用程序角色管理、访问权限设置、新用户注册及修改个人资料等功能。在实现本任务过程中，要着重掌握 ASP.net 配置工具的应用、CreateUserWizard 控件的应用、MembershipUser 类及 Membership 类的应用。

1.7.8 练习题

1. 查阅 MSDN，完成自定义 CreateUserWizard 控件，为注册用户添加城市与国家两项信息。
2. 完成自定义用户注册信息的修改。

1.8 前台购物车管理

学习目标

- 掌握购物车管理功能模块的技术分析和实现方法
- 掌握 .net Transaction-SQL 事务处理机制
- 掌握泛型集合类 Dictionary 的应用方法
- 掌握如何使用 Profiles 实现个性化配置
- 建议学时：24 学时

1.8.1 任务名称：前台购物车管理

1.8.2 任务描述

以网上购书商务网站会员资格登录的用户在浏览某种图书详细信息的过程中，如果想购买该种图书，可以单击图书详情下方的“放入购物车”链接，即可将该图书的信息添加到购物车中；同时，用户可以通过单击导航栏中的“我的购物车”进入该用户自己的购物车管理界面，从而对购物车中的图书信息进行查看、编辑、移除等操作。只有当用户进行提交生成订单的操作，用户购物车中的图书信息才会被清空。前台购物车管理功能的实现是网上购书商务网站的关键，因为购物车中的图书信息是用户个性化选择的结果，同时又是生成订单与结算支付的来源与依据。

前台购物车管理包括的主要功能有：

- ① 放入图书到购物车；
- ② 查看购物车中的图书信息；
- ③ 修改购物车中的图书数量，并重新计算总价格；
- ④ 移除购物车中的图书；
- ⑤ 清空购物车中的全部图书。

1.8.3 任务分析

如前面章节所述，本系统架构采用标准的三层架构，前台购物车管理功能的内部设计也主要为数据访问层的购物车数据访问类、业务逻辑层的购物车业务逻辑类及表示层的购物车管理页，而这三层又都可以调用购物车实体类。

在实现前台购物车管理功能时要解决好三个主要问题：

(1) 如何实现每个用户拥有自己的购物车？

每个用户拥有自己的个性化购物车，并且只要用户不提交生成订单，购物车中的数据就要为用户一直保留，依据这个业务特点，在 ASP.NET 3.5 架构下，可以采用个性化配置(Profile)来实现，Profile 可以自动在多个 Web 应用程序的访问之间存储用户信息。一个 User Profile 中可以存储各种类型的信息，这些信息既可以是简单的 string 和 integer 类型，也可以是复杂的自定义类型。例如，可以存储用户的姓名、购物车、用户属性或网站使用情况统计等。

Profile 对象与 Session 对象十分相似，但是功能更强。与 Session 相似的地方在于，Profile 也是相对于一个特定用户的，也就是说，每个 Web 应用程序的用户都有他们其的 Profile 对象。与 Session 不同的是，Profile 对象是持久对象。如果向 Session 中添加一个项，在离开网站时，该项就会消失。而 Profile 则完全不同，它为每个用户存储配置的信息是强类型，能够长期保存，并且它还支持匿名用户。

(2) 如何确定购物车中图书存放的数据结构？

向购物车中放入图书的实质是增加一个（图书书号，图书信息实例）的（键，值）对，要实现该结构最好采用泛型集合类 Dictionary<TKey, TValue>，所以使用 Dictionary 集合类来存放用户购物车中的图书信息。

(3) 如何保持购物车中的图书信息与数据库中存放购物车图书信息的数据表的一致性？

用户可以通过单击导航栏中的“我的购物车”进入自己的购物车管理界面，从而对购物车中的图书信息进行查看、编辑、移除等操作。当用户对前台购物车中的图书信息做了更改，则对数据库中存放购物车图书信息的数据表也要做相应的改变，这时需要将数据表中的原有数据删除，然后插入最新的购物车中的图书信息，对这一数据表所做的这两项操作一定要符合事务机制。

下面按照分析的思路、方法，由购物车实体类的定义到最后购物车管理页面的实现完成前台购物车管理任务的实施，并且对在实施任务过程中所用到的相关技能、知识给予详解。

1.8.4 Model层：购物车实体类CartInfo类的实现

程序开发步骤如下：

① 在 Model 类库中添加新项，选择类，取名为 CartInfo，定义所属命名空间为 namespace BookShop.Model，并标记为可序列化[Serializable]，访问修饰符设为 public。

② 主要程序代码：购物车实体类包含 4 个内部变量，分别对应数据库中购物车 Cart 表中除 UserId 之外的 4 个放入购物车中的图书信息字段。

购物车实体类 CartInfo.cs 代码如下：

```
...略
namespace BookShop.Model
{
    [Serializable]
```

```

public class CartInfo
{
    // 内部变量
    private string bookId;                //图书书号
    private string itemName;              //图书详情名称
    private decimal price;                 //售出单价
    private int quantity = 1;              //购买数量，默认值为 1

    /// <summary>
    /// 默认构造函数
    /// </summary>
    public CartInfo() { }

    /// <summary>
    /// 带参数的构造函数
    /// </summary>
    /// <param name="name">图书详情名称</param>
    /// <param name="qty">购买数量</param>
    /// <param name="price">售出单价</param>
    /// <param name="bookId">图书书号</param>
    public CartInfo(string bookId, string name, decimal price, int qty)
    {
        this.itemName = name;
        this.quantity = qty;
        this.price = price;

        this.BookId = bookId;
    }

    /// <summary>
    /// 公共属性
    /// </summary>
    public int Quantity                     //购买数量
    {
        get { return quantity; }
        set { quantity = value; }
    }

    public decimal Subtotal                 //单项总价格
    {
        get { return (decimal)(this.quantity * this.price); }
    }

    public string ItemName                  //图书详情名称
    {
        get { return itemName; }
        set { itemName = value; }
    }

    public decimal Price                    //售出单价
    {
        get { return price; }
    }
}

```

```

        set { price = value; }
    }
    public string BookId //图书书号
    {
        get { return bookId; }
        set { bookId = value; }
    }
}

```

1.8.5 DAL层：购物车数据访问类CartAccess类的实现

在 DAL 类库中添加新类——购物车数据访问类 **CartAccess**，所属命名空间为 **namespace BookShop.DAL**，访问权限为 **public**。该类的功能是对购物车 **Cart** 表进行数据访问，它包含两个主要的方法。使用 **GetCartItems** 方法能够通过登录的用户名称获得该用户保存在 **Cart** 表中的所要购买的图书信息；使用 **SetCartItems** 方法设置 **Cart** 表中的数据，如果用户在前台清空了购物车，它就会删除该用户在 **Cart** 表中的记录，如果用户改变了购物车中的图书购买信息，则它通过删除该用户在 **Cart** 表中的原有记录，然后再添加更新后的图书信息来保持与用户在购物车管理页所做操作的一致性。

1. 相关知识：事务

购物车数据访问类的 **SetCartItems** 方法在设置 **Cart** 数据表时要先后执行两个操作，首先是删除该用户原有的记录，然后添加最新的购物车中现存的图书记录。这两个操作必须保持一致性，也就是说或者两个操作都完成，或者两个都不做，这就要用到事务。

事务是作为一个逻辑工作单元执行的一系列操作，具有 **ACID**（原子性、一致性、隔离性、持久性）属性。事务启动之后，这一系列操作就必须全部成功提交，如果在执行过程中出现错误，事务启动之后对数据所作的所有修改将会回滚，将信息恢复到操作之前的状态，确保数据的安全。

由于本系统采用 **SQL Server2005** 作为数据库，所以使用 **SqlTransaction** 类在 **SQL Server** 数据库中处理 **Transact-SQL** 事务。该类所属的命名空间为 **System.Data.SqlClient**。

（1）创建 **SqlTransaction** 对象

SqlTransaction 类没有公共构造函数，不能用 **SqlTransaction myTran=new SqlTransaction();** 来新建一个 **SqlTransaction** 类型的变量，而是需要使用 **SqlConnection** 类的 **BeginTransaction();** 方法：

```

SqlTransaction trans;
trans=myConn.BeginTransaction(); //myConn 为 SqlConnection 类对象

```

该方法返回一个 **SqlTransaction** 类型的对象。在调用 **BeginTransaction()** 方法以后，所有基于该数据连接对象的 **SQL** 语句执行动作都将被认为是事务 **trans** 的一部分。同时，也可以在该方法的参数中指定事务隔离级别和事务名称，如：

```

SqlTransaction trans;
trans=myConn.BeginTransaction(IsolationLevel.ReadCommitted,"SampleTransaction");

```

（2）事务的提交方式

使用（1）里创建的 trans 对象，则为：

```
trans.Commit();
```

该语句执行后，事务针对数据库所做的一系列操作将生效，并且为数据库事务的持久性机制所保持——即使系统在这以后发生致命错误，该事务对数据库的影响也不会消失。使用 Try/Catch 进行异常处理，捕获异常则令事务发生回滚。

（3）事务的回滚

使用（1）里创建的 trans 对象，则为：

```
trans.Rollback();
```

该语句执行后，将撤销 trans 这个事务启动之后对数据表中数据所作的所有修改。

2. 在CartAccess类中使用事务的准备工作

在 CVITBOOKSHOP 数据库中分别创建两个存储过程。

存储过程 DeleteCartItems 实现依据用户名称删除 Cart 表中该用户的购物车图书记录：

```
CREATE PROCEDURE [dbo].[DeleteCartItems]
(
    @UserName varchar(256)
)
AS
Delete from Cart where UserName=@UserName
```

存储过程 InsertCart 实现向 Cart 表中添加新的购物车图书记录：

```
Create PROCEDURE [dbo].[InsertCartItems]
(
    @UserName varchar(256),
    @BookId varchar(10),
    @ItemName varchar(80),
    @Price decimal(10,2),
    @Quantity int
)
AS
insert into Cart values(@UserName,@BookId,@ItemName,@Price,@Quantity)
```

在自定义数据库连接类 DataBase 中添加 GetConnection() 方法，获得一个打开的 SqlConnection 对象：

```
public SqlConnection GetConnection()
{
    this.Open();                //初始化连接并打开连接
    return connection;          //返回连接
}
```

初始化 SqlCommand 对象：

```
public void PrepareCommand(SqlCommand cmd, SqlConnection conn, SqlTransaction trans,
```

```

CommandType ct, string cmdTxt, SqlParameter[] cmdParms)
{
    cmd.Connection = conn;
    cmd.Transaction = trans;           //把传入的事务对象和 command 关联起来
    cmd.CommandText = cmdTxt;
    cmd.CommandType = ct;
    if (cmdParms != null)             //设置参数
    {
        foreach (SqlParameter parm in cmdParms)
        {
            cmd.Parameters.Add(parm);
        }
    }
}

```

编写 ExcuteNonQuery 方法，传入事务对象参数：

```

public void ExcuteNonQuery(SqlTransaction trans, CommandType ct, string cmdTxt, SqlParameter[]
cmdParms)
{
    SqlCommand cmd = new SqlCommand();
    this.Preparecommand(cmd, trans.Connection, trans, ct, cmdTxt, cmdParms);
    cmd.ExecuteNonQuery();
    cmd.Parameters.Clear();
}

```

3. 在 CartAccess 类的 SetCartItems 方法中使用事务

CartAccess 类的 SetCartItems 方法代码清单如下：

```

/// <summary>
/// 设置 Cart 表数据
/// </summary>
/// <param name="userid">用户名</param>
/// <param name="cartItems">购物车中图书集合</param>
public void SetCartItems(string userid, ICollection<CartInfo> cartItems)
{
    SqlParameter[] delSp = { new SqlParameter("@UserName", SqlDbType.VarChar) };
    delSp[0].Value = username;
    DataBase db = new DataBase();           //创建自定义数据库连接类实例
    SqlConnection conn = db.GetConnection(); //获得已初始化并被打开的数据库连接
    SqlTransaction trans = conn.BeginTransaction(); //基于数据连接开始一个事务
    if (cartItems.Count > 0)                //判断用户购物车中是否存在选择的
    图书
    {
        try
        {
            //属于事务 trans 中的操作删除用户购物车中的原有记录

```

```

db.ExcuteNonQuery(trans, CommandType.StoredProcedure, "DeleteCartItems", delSp);
//定义添加用户购物车中选定的图书信息所需参数
SqlParameter[] insertSp = {new SqlParameter("@UserName", SqlDbType.VarChar),
new SqlParameter("@BookId", SqlDbType.VarChar),
new SqlParameter("@ItemName", SqlDbType.VarChar),
new SqlParameter("@Price", SqlDbType.Decimal),
new SqlParameter("Quantity", SqlDbType.Int) };
        insertSp[0].Value =username;
//逐条为参数赋值，并完成向 Cart 表添加用户购物车中选定的图书记录
        foreach (CartItem cartItem in cartItems)
        {
            insertSp[1].Value = cartItem.BookId;
            insertSp[2].Value = cartItem.ItemName;
            insertSp[3].Value = cartItem.Price;
insertSp[4].Value = cartItem.Quantity;
//属于事务 trans 中的操作添加用户购物车中选定的图书记录

db.ExcuteNonQuery(trans, CommandType.StoredProcedure, "InsertCartItems", insertSp);
                trans.Commit();                //事务提交
        }
    }
    catch(Exception ex)
    {
        trans.Rollback();                //事务回滚
        throw ex;
    }
    finally
    {
        conn.Close();
    }
}
else
{
    db.ExcuteNonQuery(CommandType.StoredProcedure, "DeleteCartItems ", delSp);
}
}

```

4. 购物车数据访问类CartAccess中获得用户购物车中图书信息的GetCartItems方法的实现

程序开发步骤如下。

- ① 在 CVITBOOKSHOP 数据库中创建存储过程 GetCartItems:

```

CREATE PROCEDURE [dbo].[GetCartItems]
@Username varchar(256),

AS
SELECT dbo.Cart.BookId, dbo.Cart.ItemName, dbo.Cart.Price, dbo.Cart.Quantity

```


② 在 DAL 类库中的 CartAccess 类中编写如下代码。

购物车数据访问类 CartAccess 代码：

```
namespace BookShop.DAL
{
    public class CartAccess
    {
        /// <summary>
        /// 获取 Cart 表中购物车图书记录
        /// </summary>
        /// <param name="userid">用户名称</param>
        public IList<CartInfo> GetCartItems(string username)
        {
            //定义获取用户购物车信息存储过程所需要的参数
            SqlParameter[] sp = { new SqlParameter("@UserName", SqlDbType.VarChar) };
            //给参数赋值
            sp[0].Value = username;
            DataBase db = new DataBase();
            //执行存储过程
            SqlDataReader sdr = db.ExcuteDataReader(CommandType.StoredProcedure, "GetCartItems", sp);
            IList<CartInfo> cartItems = new List<CartInfo>();
            while (sdr.Read())
            {
                //用带参构造方法构造购物车实体类对象
                CartInfo cartItem = new CartInfo(sdr.GetString(0), sdr.GetString(1), sdr.GetDecimal(2), sdr.GetInt32(3));
                //添加到 IList <CartInfo> 中
                cartItems.Add(cartItem);
            }
            sdr.Close();
            //返回查询信息
            return cartItems;
        }
        public void SetCartItems(string userid, ICollection<CartInfo> cartItems)
        {
            .....代码略，详见在 SetCartItems 方法中使用事务
        }
    }
}
```

1.8.6 BLL层：购物车业务逻辑类CartManager类的实现

在 BLL 类库中添加新类，命名为 CartManager，定义所属命名空间为 namespace BookShop.BLL，标记为可序列化[Serializable]，访问权限为 public。该类负责用户放入图书到购物车、修改购物车信息、移除购物车中图书项、清空购物车等业务处理，对数据库中 Cart 表的操作，该类会调用购物车数据访问类 CartAccess 来完成。

前面介绍过，向购物车中放入图书的实质是增加一个（图书书号，图书信息实例）的（键，

值）对，实现该结构从类型安全与性能考虑最好采用泛型集合类 Dictionary<TKey, TValue>，所以设计购物车业务逻辑类 CartManager 包含的内部成员为一个 Dictionary<string, CartInfo> 集合，用它来存放购物车中的图书信息。

1. 相关技能、知识：Dictionary<TKey, TValue>集合类

该泛型集合类所属命名空间为 System.Collections.Generic，使用它可以提供更高的类型安全性，获得更优越的性能，能够避免非泛型集合重复的装箱和拆箱操作。

很多非泛型集合类都有对应的泛型集合类，表 1-37 是常用的非泛型集合类及其对应的泛型集合类列表。

表 1-37 非泛型集合类及其对应的泛型集合类列表

非泛型集合类	泛型集合类
ArrayList	List<T>
HashTable	Dictionary<Tkey, TValue>
Queue	Queue<T>
Stack	Stack<T>
SortedList	SortedList<T>

用得比较多的非泛型集合类主要有 ArrayList 类和 HashTable 类，而在使用它们存储将要写入到数据库或者返回的信息时，要不断地进行类型的转化，增加了系统装箱和拆箱的负担。如果操纵的数据类型相对确定，则用 Dictionary<TKey, TValue> 集合类存储数据就方便多了。在电子商务网上购书管理系统中存储用户的购物车信息（图书书号，对应的图书对象）时，完全可以用 Dictionary<string, CartInfo> 存储，而不需要任何的类型转化。Dictionary<TKey, TValue>集合类的主要属性如表 1-38 所示，主要方法如表 1-39 所示。

表 1-38 Dictionary<TKey, TValue>集合类的主要属性

名 称	说 明
Comparer	获取用于确定 Dictionary 中的键是否相等的 IEqualityComparer
Count	获取包含在 Dictionary 中的键/值对的数目
Item	获取或设置与指定的键相关联的值
Keys	获取包含 Dictionary 中的键的集合
Values	获取包含 Dictionary 中的值的集合

表 1-39 Dictionary<TKey, TValue>集合类的主要方法

名 称	说 明
Add	将指定的键和值添加到 Dictionary 中
Clear	从 Dictionary 中移除所有的键和值
ContainsKey	确定 Dictionary 是否包含指定的键
Equals	已重载。确定两个 Object 实例是否相等（从 Object 继承）
GetEnumerator	返回循环访问 Dictionary 的枚举数
GetHashCode	用做特定类型的哈希函数。GetHashCode 适合在哈希算法和数据结构（如哈希表）中使用（从 Object 继承）

名 称	说 明
GetObjectData	实现 System.Runtime.Serialization.ISerializable 接口，并返回序列化 Dictionary 实例所需的数据
GetType	获取当前实例的 Type（从 Object 继承）
OnDeserialization	实现 System.Runtime.Serialization.ISerializable 接口，并在完成反序列化之后引发反序列化事件
ReferenceEquals	确定指定的 Object 实例是否是相同的实例（从 Object 继承）
Remove	从 Dictionary 中移除所指定的键的值
ToString	返回表示当前 Object 的 String（从 Object 继承）
TryGetValue	获取与指定的键相关联的值

2. Dictionary<TKey, TValue>集合类在购物车业务逻辑类CartManager中的应用

```

namespace BookShop.BLL
{
    [Serializable]
    public class CartManager
    {
        //定义 Dictionary<string, CartInfo>对象 cartItems 存放购物车图书信息
        private Dictionary<string, CartInfo> cartItems = new Dictionary<string, CartInfo>();
        private static readonly CartAccess dal = new CartAccess();

        public decimal Total
        {
            get
            {
                decimal total = 0;
                //foreach 包含在 Dictionary 中值的集合的 CartInfo 对象
                foreach (CartInfo cartItem in cartItems.Values)
                {
                    total = total + cartItem.Price * cartItem.Quantity;
                }
                return total;
            }
        }
        public int count//获取商品数目
        {
            get
            {
                //获取包含在 Dictionary 中的键/值对的数目
                return cartItems.Count;
            }
        }
        public ICollection<CartInfo> CartItems
        {
            get

```

```

{
//获取包含在 Dictionary 中的值的集合
return cartItems.Values;
}
}

public void SetQuantity(string bookId, int qty)
{
//通过键 bookId 获取对应的值——书号为 bookId 的 CartInfo 对象,设置其 Quantity 属性值
cartItems[bookId].Quantity = qty;
}
/// <summary>
/// 传入图书书号向购物车中添加图书
/// </summary>
/// <param name="bookId"></param>
public void Add(string bookId)
{
CartInfo cartItem;
//在 Dictionary 中获取与指定的键(bookId)相关联的值,找到返回真,否则为假,并返回值
if (!cartItems.TryGetValue(bookId, out cartItem))
{
ItemManager item = new ItemManager();
ItemInfo data = item.GetItem(bookId);
if (data != null)
{
CartInfo newcartitem=new CartInfo(data.BookId, data.ItemName, data.ListPrice, 1);
//将指定的键和值添加到 Dictionary 中
cartItems.Add(bookId, newcartitem);
}
}
else
{
//Dictionary 中如果已存在该书,则只将 CartInfo 对象的 Quantity 属性值加 1
cartItem.Quantity = cartItem.Quantity + 1;
}
}
/// <summary>
/// 传入 CartInfo 对象向购物车中添加图书
/// </summary>
/// <param name="cartItem"></param>
public void Add(CartInfo cartItem)
{
CartInfo item;
if (!cartItems.TryGetValue(cartItem.BookId,out item))
{
cartItems.Add(cartItem.BookId, cartItem);
}
}
}

```

```

        }
        else
            item.Quantity+=1;
    }
    /// <summary>
    /// 按图书书号从购物车中移除图书
    /// </summary>
    /// <param name="bookId">图书书号</param>
    public void Remove(string bookId)
    {
        //从 Dictionary 中移除所指定的键的值
        cartItems.Remove(bookId);
    }
    /// <summary>
    /// 清空购物车中所有的图书
    /// </summary>
    /// <param name="bookId">图书书号</param>
    public void Clear()
    {
        //从 Dictionary 中移除所有的键和值
        cartItems.Clear();
    }
    /// <summary>
    /// 调用 CartAccess 成员获取购物车 Cart 表中用户购物车中的图书记录
    /// </summary>
    /// <param name="userid">用户名</param>
    public void GetCartItems(string userid)
    {
        foreach (CartItem cartItem in dal.GetCartItems(userid))
        {
            this.Add(cartItem);
        }
    }
    /// <summary>
    /// 调用 CartAccess 成员设置购物车 Cart 表中用户购物车数据
    /// <param name="userid">用户名</param>
    /// </summary>
    public void SetCartItems(string userid)
    {
        dal.SetCartItems(userid, this.CartItems);
    }

    /// <summary>
    /// 把购物车中的图书信息转换成订单明细信息
    /// </summary>
    /// <returns>订单明细数组</returns>
    public LineItemInfo[] GetOrderLineItems()

```

```

{
    LineItemInfo[] orderLineItems = new LineItemInfo[cartItems.Count];
    int lineNum = 0;
    foreach (CartItem item in cartItems.Values)
        orderLineItems[lineNum] = new LineItemInfo(++lineNum,item.BookId,item.Quantity,item.Price);

    }
    return orderLineItems;
    }
}
}

```

1.8.7 购物车表示层代码的实现

前台购物车管理在表示层包含购物车管理页面 ShoppingCart.aspx 及其代码文件 ShoppingCart.aspx.cs。

1. 购物车管理页ShoppingCart.aspx设计步骤

① 在 WebUI 网站中新建一个 Web 窗体，命名为 ShoppingCart.aspx，并选择 Master Page.master 为母版页。

② 在页面中添加一个 div，为整个页面布局。从“工具箱”的“数据”选项卡中拖放一个 Label 控件，显示用户购物车中存在的记录条目数信息，拖放一个 Repeater 控件，主要显示购物车中的图书信息，拖放一个 Panel 控件，显示用户购物车图书的总价格信息，拖放两个 HyperLink 控件，用来完成“继续购物”与“生成订单”的操作链接。在 Repeater 控件的模板中添加一个 Table，作为购物车信息显示的布局，并依次向其中加入一个 Button 控件和一个 TextBox 控件。在 Panel 控件中加入两个 Label 控件和一个 Button 控件。

通过属性窗口设置控件属性。

页面中各个控件的属性设置及其用途如表 1-40 所示。

表 1-40 ShoppingCart.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
Label	lblShow	Text 属性设置为“您购物车中的商品”	显示说明文字
	lblTotal	Text 属性设置为"0.00Y"	显示购物车图书总价格
Repeater	repShoppingCart		显示购物车中的图书详细购买信息
ImageButton	btnDelete	CommandName 属性设置为"Del" CommandArgument 属性设置为'<%# Eval("BookId") %>'	执行从购物车中移除图书的操作
TextBox	txtQty	Text 属 性 设 置 为 '<%# Eval("Quantity") %>'	显示购买图书的数量
Button	btnTotal	Text 属性设置为“结算”	执行计算购物车总价格的操作
	btnClear	Text 属性设置为“清空购物车”	执行清空购物车的操作
Panel	plhTotal	Visible 属性设置为“false”	购物车总价格显示区域

控 件 类 型	控 件 名 称	主要属性设置	用 途
HyperLink	hlDefault	NavigateUrl 属 性 设 置 为 "~/Default.aspx"	执行继续购物的网页定向
	hlOrder	NavigateUrl 属 性 设 置 为 "~/CheckOut.aspx"	执行生成订单的网页定向

ShoppingCart.aspx 页面的 HTML 代码如下：

```
<%@PageLanguage="C#"MasterPageFile=~\MasterPage.master"AutoEventWireup="true"CodeFile=
"Cart.aspx.cs" Inherits="Cart" Title="我的购物车" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<div>
<table width="500">
<tr><td> <asp:Label ID="lblShow" runat="server" Text="您购物车中的商品:"></asp:Label></td>
<td align="right"><asp:Button ID="btnClear" runat="server" Text="清空购物车" /></td></tr>
</table>
<asp:Repeater ID="repShoppingCart" runat="server" onitemcommand=" repShoppingCart
_ItemCommand">
<HeaderTemplate>
<table width="500px" cellpadding="3" cellspacing="0" rules="none" align="center">
<tr bgcolor="#F7FAFE">
<th>移除</th>
<th>图书名称</th>
<th>数量</th>
<th>单价</th> </tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td align="center">
<asp:ImageButtonID="btnDelete"runat="server"CommandName="Del"
CommandArgument='<%=#Eval("BookId")%>'ImageUrl=~\images\gif-0082.gifCausesValidation="false
" />
</td>
<td align="center">
<a href='<%=#"BookItem.aspx?BookId="+Eval("BookId")%>' width="80%"><%=# Eval("ItemName")
%></a>
</td>
<td align="center">
<asp:TextBoxID="txtQty"runat="server"Text='<%=#Eval("Quantity")%>'Width="40px"></asp:TextBox>
</td>
<td align="center" width="20%">
<%=# Eval("Price", "{0:c}") %>
</td>
</tr>
```

```

</ItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>
    <asp:Panel ID="plhTotal" runat="server" Visible="False">
    <table>
    <tr>
    <td>总价格:</td>
    <td>
        <asp:Button ID="btnTotal" runat="server" Text="结算" CausesValidation="False"
            onclick="btnTotal_Click" />
    </td>
    <td>
        <asp:Label ID="lblTotal" runat="server" Text="Label"></asp:Label>
    </td>
    <td>&nbsp;  </td>
    </tr>
    <tr>
    <td> &nbsp;  </td>
    <td> &nbsp;  </td>
    </tr>
    <td>
        <asp:HyperLink ID="hlDefault" runat="server" NavigateUrl="~/Default.aspx">继续购物
</asp:HyperLink>
    </td>
    <td>
        <asp:HyperLink ID="hlOrder" runat="server" NavigateUrl="~/CheckOut.aspx">生成订单
</asp:HyperLink>
    </td>
    </tr>
    </table>
    </asp:Panel>
</div>
</asp:Content>

```

2. 代码文件ShoppingCart.aspx.cs的实现

为了满足每个用户拥有自己的个性化购物车，并且只要用户不提交生成订单，购物车中的数据就要为用户一直保留的业务特点，在 ASP.NET 3.5 架构下，采用个性化配置（Profile）来实现，Profile 可以自动在多个 Web 应用程序的访问之间存储用户信息。一个 UserProfile 中可以存储各种类型的信息，这些信息既可以是简单的 string 和 integer 类型，也可以是复杂的自定义类型。

3. 相关技能与知识：Profile（个性化配置）

Profile 对象与 Session 对象十分相似，但是功能更强大。与 Session 相似的地方在于，Profile 是相对于一个特定用户的，也就是说，每个 Web 应用程序的用户都有其自己的 Profile 对象。与 Session 不同的是，Profile 对象是持久对象。如果向 Session 中添加一个项，在离开网站时，

该项就会消失。而 Profile 则完全不同，当修改 Profile 的状态时，修改在多个访问之间均有效。另外，Profile 是强类型的，而 Session 对象仅仅是一个项集合而已。使用强类型的好处在于可以在 Microsoft Visual Web Developer 中使用智能感知技术，当输入 Profile 和一个点的时候，智能感知会弹出已经定义过的 Profile 属性列表。

Profile 使用 provider 模式来存储信息，默认情况下，UserProfile 的内容会保存在 SQL Server Express 数据库中，该数据库位于网站的 App_Data 目录下。也可以自定义其他数据提供者（data provider）来存储信息，如完整版的 SQL Server 中的一个数据库或者一个 Oracle 数据库。

（1）定义 UserProfile

可以在 machine.config 中、也可以在 web.config 中定义一个 UserProfile，不能在应用程序的二级目录中创建一个包含 Profile 节的 web.config 文件，这意味着无法在一个应用程序中定义两个以上的 Profile。下面在 web.config 文件中定义一个简单的 Profile 的实例，该 Profile 有 3 个属性：UserName、Address 和 PageHits。

在 web.config 的 <system.web> 配置节中添加 <profile> 配置节：

```
<configuration>
<system.web>
  <profile>
    <property>
      <add name="UserName" defaultValue="song" allowAnonymous="true" />
      <add name="Address"  defaultValue="changchun" allowAnonymous="true" />
      <add name="PageHits" type="Int32" allowAnonymous="true"/>
    </properties>
  </profile>
</system.web>
</configuration>
```

默认的 Profile 属性类型是 System.String 类型。由于没有为 UserName 和 Address 这两个 Profile 属性增加 type 特性，因此系统默认它们是 string 类型，而 PageHits 属性则指定了 type 特性为 Int32，因此该 Profile 属性可用于表示一个整型值。

注意：UserName 和 Address 属性都有 defaultValue 特性。可以为简单的数据类型设置 defaultValue 特性，但不能为复杂的类型设置 defaultValue 特性。

当定义好一个 Profile 之后，系统会自动在下一次页面被调用时生成一个与该 Profile 相对应的类。这个类会被保存在“Temporary ASP.NET Files Directory”目录（该目录也用于存放用来动态生成页面的类），可以使用 HttpContext 的 Profile 属性（Property）调用该类。任何在 web.config 中定义的 Profile 属性都会 Profile 对象中呈现。

（2）使用 Profile 来持久化保存用户信息。

下列 Test.aspx 代码能够显示 UserName、Address、PageHits 3 个属性的值，同时它还包含了一个用于修改这 3 个属性的表单（form）。在 Page_Load 中更新 PageHits 的值，这意味着每一次刷新页面，PageHits 的值都会改变。

Test.aspx 代码如下：

```
<%@ Page Language="C#" %>
<script runat="server">
void Page_Load()
```

```

    {
        Profile.PageHits ++;
    }

void UpdateProfile(Object s, EventArgs e)
{
    Profile.UserName = txtUserName.Text;
    Profile.Address = txtAddress.Text;
}
</script>
<html>
<head>
    <title>Test</title>
</head>
<body>
    <form id="form1" runat="server">
        <b>用户名称: </b> <%= Profile.UserName %>
    <br />
    <b>用户地址: </b> <%= Profile.Address %>
    <br />
    <b>浏览次数: </b> <%= Profile.PageHits %>
    <hr />
    <b>用户名称: </b>
    <asp: TextBox ID="txtUserName" Runat="Server" />
    <br />
    <b>用户地址: </b>
    <asp: TextBox ID="txtAddress" Runat="Server" />
    <br />
    <asp: Button ID="btnUpdate"
        Text="更新个性化配置"
        onClick="UpdateProfile"
        Runat="server" />
    </form>
</body>
</html>

```

如果多次访问 Test.aspx 页面，则 PageHits 每一次都加一，并且 PageHits 属性会一直保留其值。所以可以看出，Profile 为每个用户自动保存一个副本。

4. 使用Profile 实现每个用户拥有自己的购物车

在电子商务网上购书 WebUI 网站的 Web.config 文件中添加<profile>配置节：

```

<profile>
<properties>
    <addname="ShoppingCart"type="BookShop.BLL.CartManager" allowAnonymous="false" />
</properties>
</profile>

```

Profile 属性的名称是 ShoppingCart，它的类型是复杂类型购物车业务逻辑类 CartManager，

注意在配置的时候一定要指出它所属的命名空间。Profile 会为每个用户保存 ShoppingCart 的值，从而 CartManager 对象中用来存放购物车图书信息的成员 Dictionary<string, CartIno>集合类对象 cartItems 也将为每个用户保存。每个用户还可以使用 CartManager 对象的放入图书到购物车、移除图书等方法来操作自己的购物车，从而实现每个用户拥有自己的购物车。

ShoppingCart.aspx.cs 代码如下：

```
public partial class ShoppingCart : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            /**使用 Profile 属性 ShoppingCart,其类型为 CartManager,调用 GetCartItems 方法获得
            用户购物车数据,Profile.UserName 获得与该 Profile 关联的用户名**/
            Profile.ShoppingCart.GetCartItems(Profile.UserName);
            string BookId = Request.QueryString["BookId"];
            if (!string.IsNullOrEmpty(BookId))
            {
                if (Session["flag"].ToString().Equals(""))
                {
                    //使用 Profile 属性 ShoppingCart,调用 Add 方法添加图书到购物车中
                    Profile.ShoppingCart.Add(BookId);
                    Session["flag"] = BookId;
                }
            }
            //绑定数据源到 repShoppingCart 控件
            BindCart(profile.ShoppingCart);
            /**使用 Profile 属性 ShoppingCart,调用 SetCartItems 方法设置用户购物车数据,保存用户购物车信息
            的变化**/
            Profile.ShoppingCart.SetCartItems(Profile.UserName);
        }

        //计算购物车总价格
        protected void btnTotal_Click(object sender, EventArgs e)
        {
            Profile.ShoppingCart.GetCartItems(Profile.UserName);
            TextBox txtQty;
            ImageButton btnDel;
            foreach (RepeaterItem row in repShoppingCart.Items)
            {
                txtQty = (TextBox)row.FindControl("txtQty");
                btnDel = (ImageButton)row.FindControl("btnDelete");
                int qty = Int32.Parse(txtQty.Text);
                if (qty>0)
                {
                    //使用 Profile 属性 ShoppingCart, 调用 Add 方法添加图书到购物车中
```

```

        Profile.ShoppingCart.SetQuantity(btnDel.CommandArgument, qty);
    }

    else
    {
        if (qty==0)
        {
            //使用 Profile 属性 ShoppingCart, 调用 Remove 方法移除图书
            Profile.ShoppingCart.Remove(btnDel.CommandArgument);
        }
    }
    BindCart(Profile.ShoppingCart);
    Profile.ShoppingCart.SetCartItems(Profile.UserName);
}

//绑定数据源到 repShoppingCart 控件
protected void BindCart(CartManager cm)
{
    //判断购物车中是否有图书信息
    if (cm.Count > 0)
    {
        repShoppingCart.DataSource = cm.CartItems;
        repShoppingCart.DataBind();
        //以货币格式显示总价格
        lblTotal.Text = cm.Total.ToString("c");
        //设置显示总价格区域可见
        plhTotal.Visible = true;
    }
    else
    {
        repShoppingCart.Visible = false;
        plhTotal.Visible = false;
        lblShow.Text = "你的购物车中尚没有商品!";
        plhTotal.Visible = false;
    }
}

//处理在 repShoppingCart 中单击移除按钮事件
protected void repShoppingCart_ItemCommand(object source, RepeaterCommandEventArgs e)
{
    Profile.ShoppingCart.GetCartItems(Profile.UserName);
    switch (e.CommandName.ToString())
    {
        //依据 CommandName 属性识别事件源
        case "Del": profile.ShoppingCart.Remove(e.CommandArgument.ToString());
        break;
    }
}

```

```

        default:
            break;
    }
    Profile.ShoppingCart.SetCartItems(Profile.UserName);
    BindCart(Profile.ShoppingCart);
}
}

//执行清空购物车操作
protected void btnClear_Click(object sender, EventArgs e)
{
    //使用 Profile 属性 ShoppingCart, 调用 Clear 方法清空购物车
    Profile.ShoppingCart.Clear();
    Profile.ShoppingCart.SetCartItems(Profile.UserName);
}

```

购物车管理页面 ShoppingCart.aspx 运行效果图如图 1-62 所示。

5. 购物车管理页面ShoppingCart.aspx运行效果图



图 1-62 购物车管理页面 ShoppingCart.aspx 运行效果图

1.8.8 任务小结

在完成前台购物车管理的任务中遵循三层标准架构，主要实现了购物车数据访问类、购物车业务逻辑类、我的购物车管理页面及代码等，在对数据访问时应用了事务机制，在存放购物车数据时应用了泛型集合类 **Dictionary**，在表示层应用了 **Profile** 个性化配置。正是依靠对上述知识的合理运用，很好地解决了在任务分析过程中所提出的 3 个关键问题。

在本任务对 **Profile** 的应用中，使用的是默认的提供程序，如果想更灵活地应用 **Profile**，可以自定义 **ProfileProvider** 配置提供程序。

ProfileProvider: ASP.NET 为使用自定义配置文件提供程序提供配置文件服务而实现的协定。可以在下列情况下创建自定义配置文件提供程序。

① 要将配置文件信息存储在 .NET Framework 附带的配置文件提供程序不支持的数据源中，例如，FoxPro 数据库、Oracle 数据库或其他数据存储。

② 您需要使用与 .NET Framework 附带的提供程序所用的数据库架构不同的数据库架构来管理配置文件信息。例如，公司网络或网站的 SQL Server 数据库中已经存在的用户数据。

③ 若要实现配置文件提供程序，必须创建一个继承 **System.web.Profile.ProfileProvider** 抽象类的类。读者可根据实际情况拓展此功能。

1.8.9 练习题

1. 完善程序：.NET 中事务的使用。

```
public bool UseTran()
{
    SqlConnection myConnection = new SqlConnection("DataSource=localhost;InitialCatalog=CVITBOOKS
HOP;Integrated Security=SSPI;");
    myConnection.Open();
    SqlTransaction myTrans = _____; //开始一个事务
    SqlCommand myCommand = new SqlCommand();
    myCommand.Transaction = myTrans;
    Try
    {
        myCommand.CommandText = "Update Address set location='23 rain street' where userid='0001'";
        myCommand.ExecuteNonQuery();
        _____; //事务提交
    }
    return True;
}
catch(Exception e)
{
    _____; //事务回滚
}
return False;
}
finally
{
    myConnection.Close();
}
```

```
}
```

2. 完善程序：.NET 中 Dictionary 的遍历。

```
public void traverse()
{
    Dictionary<int, int> dict = _____;
    _____(10, 6);
    _____(18, 5);
    foreach(_____ in dict)
    Response.Write(string.Format("key: {0}\nvalues: {1} ", entry.Key, entry.Value));
}
```

1.9 前台订单管理实现

学习目标

- 掌握 Wizard 控件的应用
- 掌握对数据库触发器的应用
- 建议学时：16 学时

1.9.1 任务名称：前台订单管理实现

1.9.2 任务描述

会员将选定的图书放入购物车之后，单击“生成订单”链接，转至 `CheckOut.aspx` 页面，在该页面设定订单发送地址与货物接收地址，最后提交订单。会员还可以查看自己的订单。

1.9.3 任务分析

生成订单的过程需要分步骤实现两类地址的设定，所以最好采用 Wizard 控件来实现。提交订单的过程要保证事务的完整性机制，不能存在没有订单明细的订单数据，同时每一项订单明细都要有订单状态表示值。订单的查看则主要通过主从表来实现。

1.9.4 在Model类库中创建OrdersInfo、LineItemInfo等业务实体类

代码参见 1.5.5 在类库 Model 中创建业务实体类。

1.9.5 在数据库中创建存储过程与触发器

1. 创建存储过程插入订单

```
Create PROCEDURE [dbo].[InsertOrder]
@UserId varchar(256),@ShipToName varchar(80),@ShipEmail varchar(80),
@ShipAddr varchar(80),@ShipCity varchar(80),@ShipState varchar(80),
@ShipZip varchar(20),@ShipCountry varchar(20),@ShipPhone varchar(80),
@BillToName varchar(80),@BillEmail varchar(80),@BillAddr varchar(80),
```

```

@BillCity varchar(80),@BillState varchar(80),@BillZip varchar(20),
@BillCountry varchar(20),@BillPhone varchar(80),@TotalPrice decimal(10,2)
AS
insert into Orders(UserId,ShipToName,ShipEmail,
ShipAddr,ShipCity,ShipState,
ShipZip,ShipCountry,ShipPhone,
BillToName,BillEmail,BillAddr,
BillCity,BillState,BillZip,
BillCountry,BillPhone,TotalPrice)
values(@UserId,@ShipToName,@ShipEmail,
@ShipAddr,@ShipCity,@ShipState,
@ShipZip,@ShipCountry,@ShipPhone,
@BillToName,@BillEmail,@BillAddr,
@BillCity,@BillState,@BillZip,
@BillCountry,@BillPhone,@TotalPrice)
return @@identity

```

2. 创建存储过程插入订单明细

```

Create PROCEDURE [dbo].[InsertLineItem]
    @OrderId int,@LineNum int,@BookId varchar(10),@Quantity int,@UnitPrice decimal(10,2)
AS
insert into LineItem
values(@OrderId,@LineNum,@BookId,@Quantity,@UnitPrice)

```

3. 创建触发器、当插入明细时向订单状态表中插入订单状态

```

Create TRIGGER [dbo].[TriLineItem]
    ON [dbo].[LineItem]
    AFTER insert
AS
declare @OrderId int
declare @LineNum int
select @OrderId=OrderId,@LineNum=LineNum from inserted
insert into OrderStatus(OrderId,LineNum)
values(@OrderId,@LineNum)

```

1.9.6 在DAL类库中创建OrderAccess类

在 DAL 类库中编辑如下代码，重新生成 DAL 类库。

```

...(略)
namespace BookShop.DAL
{
    public class OrderAccess
    {
        //插入订单
        public void InsertOrder(OrdersInfo order)
        {
            //实例化 Order 表中参数并赋值

```



```
SqlParameter[] sp = { new SqlParameter("@UserId", SqlDbType.VarChar),
    new SqlParameter("@ShipToName", SqlDbType.VarChar),
    new SqlParameter("@ShipEmail", SqlDbType.VarChar),
    new SqlParameter("@ShipAddr", SqlDbType.VarChar),
    new SqlParameter("@ShipCity", SqlDbType.VarChar),
    new SqlParameter("@ShipState", SqlDbType.VarChar),
    new SqlParameter("@ShipZip", SqlDbType.VarChar),
    new SqlParameter("@ShipCountry", SqlDbType.VarChar),
    new SqlParameter("@ShipPhone", SqlDbType.VarChar),
    new SqlParameter("@BillToName", SqlDbType.VarChar),
    new SqlParameter("@BillEmail", SqlDbType.VarChar),
    new SqlParameter("@BillAddr", SqlDbType.VarChar),
    new SqlParameter("@BillCity", SqlDbType.VarChar),
    new SqlParameter("@BillState", SqlDbType.VarChar),
    new SqlParameter("@BillZip", SqlDbType.VarChar),
    new SqlParameter("@BillCountry", SqlDbType.VarChar),
    new SqlParameter("@BillPhone", SqlDbType.VarChar),
    new SqlParameter("@TotalPrice", SqlDbType.Decimal),
    new SqlParameter("@Return", SqlDbType.Int)
};
```

//赋值

```
sp[0].Value = order.UserId;
sp[1].Value = order.ShippingAddress.Name;
sp[2].Value = order.ShippingAddress.Email;
sp[3].Value = order.ShippingAddress.Address;
sp[4].Value = order.ShippingAddress.City;
sp[5].Value = order.ShippingAddress.State;
sp[6].Value = order.ShippingAddress.Zip;
sp[7].Value = order.ShippingAddress.Country;
sp[8].Value = order.ShippingAddress.Phone;
sp[9].Value = order.BillingAddress.Name;
sp[10].Value = order.BillingAddress.Email;
sp[11].Value = order.BillingAddress.Address;
sp[12].Value = order.BillingAddress.City;
sp[13].Value = order.BillingAddress.State;
sp[14].Value = order.BillingAddress.Zip;
sp[15].Value = order.BillingAddress.Country;
sp[16].Value = order.BillingAddress.Phone;
sp[17].Value = order.TotalPrice;
sp[18].Direction = ParameterDirection.ReturnValue;
```

DataBase db = new DataBase();

SqlConnection conn = db.GetConnection();//事务必须基于数据库连接

SqlTransaction trans = conn.BeginTransaction();//开始一个事务

try

{

```

        //接收传过来的 OrderId
objectob=db.ExcuteNonQueryReturn(trans, CommandType.StoredProcedure, "InsertOrder", sp);
int id = Int32.Parse(ob.ToString());
//实例化 LineItem 表中参数
SqlParameter[] linesp = { new SqlParameter("@OrderId", SqlDbType.Int),
                           new SqlParameter("@LineNum", SqlDbType.Int),
                           new SqlParameter("@BookId", SqlDbType.VarChar),
                           new SqlParameter("@Quantity", SqlDbType.Int),
                           new SqlParameter("@UnitPrice", SqlDbType.Decimal)
                           };

//为 OrderId 赋值
linesp[0].Value = id;

//把每一个 OrderId 对应的订单明细信息插入到订单明细表中
foreach (LineItemInfo line in order.LineItems)
{
    linesp[1].Value = line.LineNum;
    linesp[2].Value = line.BookId;
    linesp[3].Value = line.Quantity;
    linesp[4].Value = line.UnitPrice;
    db.ExcuteNonQuery(trans, CommandType.StoredProcedure, "InsertLineItem", linesp);
}
trans.Commit();//事务提交，正常完成
}
catch (Exception ex)
{
    trans.Rollback();//事务回滚
    throw ex;
}
finally
{
    conn.Close();
}
}

//根据用户标识获取该用户已生成
public DataSet GetOrdersByUserId(string userId)
{
    SqlParameter[] sp = { new SqlParameter("@UserId", SqlDbType.VarChar) };
    sp[0].Value = userId;
    DataBase db = new DataBase();
    DataSet ds = new DataSet();
    SqlDataAdapter sda=db.CreateDataAdapter(CommandType.StoredProcedure,
    "GetOrders", sp, 1);
    sda.Fill(ds);
    return ds;
}

//根据订单编号获取该订单的订单明细

```

```

        public DataSet GetLineItemsByOrderId(int orderId)
        {
            SqlParameter[] sp = { new SqlParameter("@OrderId", SqlDbType.Int) };
            sp[0].Value = orderId;
            DataBase db = new DataBase();
            DataSet ds = new DataSet();
            SqlDataAdapter sda = db.CreateDataAdapter(CommandType.StoredProcedure,
"GetLineItems", sp, 1);
            sda.Fill(ds);
            return ds;
        }
    }
}

```

1.9.7 在BLL类库中创建OrderManager类

添加对 DAL 类库的引用，在 BLL 类库中编辑如下代码，重新生成 BLL 类库。

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BookShop.DAL;
using BookShop.Model;
using System.Data.SqlClient;
using System.Data;
namespace BookShop.BLL
{
    [Serializable]
    public class OrderManager
    {
        //创建私有数据成员订单数据访问类 OrderAccess 对象 dal
        public static readonly OrderAccess dal = new OrderAccess();
        //插入订单
        public void InsertOrder(OrdersInfo order)
        {
            dal.InsertOrder(order);
        }
        //依据用户标识获取该用户已生成订单
        public DataSet GetOrdersByUserId(string userId)
        {
            return dal.GetOrdersByUserId(userId);
        }
        //依据订单编号获取该订单明细
        public DataSet GetLineItemsByOrderId(int orderId)
        {
            return dal.GetLineItemsByOrderId(orderId);
        }
    }
}

```

```
}  
}
```

1.9.8 表示层CheckOut.aspx页面的实现

1. 相关知识：Wizard控件的应用

使用 Wizard 控件可以简化许多与生成一系列窗体以收集用户输入的操作关联的任务。

通过使用窗体收集用户输入是 Web 开发中一个要反复涉及的任务。用来完成某个任务的一组窗体通常称为向导。

ASP.NET Wizard 控件简化了许多与生成多个窗体及收集用户输入的操作关联的任务。Wizard 控件提供了一种简单的机制，允许轻松地生成步骤、添加新步骤或重新安排步骤。无须编写代码即可生成线性和非线性的导航，并自定义控件的用户导航。

Wizard 控件使用多个步骤来描绘用户数据输入的不同部分。该控件内的每个步骤均会给定一个 StepType，用以指示这一步骤是开始步骤、中间步骤还是完成步骤。向导可以根据需要带有任意数量的中间步骤。可以添加不同的控件（如 TextBox 或 ListBox 控件）来收集用户输入。当到达 Complete 步骤时，所有数据都可供访问。下面的代码示例演示带有两个步骤的 Wizard 控件：

```
<asp: Wizard ID="Wizard1" Runat="server">  
  <WizardSteps>  
    <asp: WizardStep Runat="server" Title="第一步">  
    </asp: WizardStep>  
    <asp: WizardStep Runat="server" Title="第二步">  
    </asp: WizardStep>  
  </WizardSteps>  
</asp: Wizard>
```

在每个步骤中都可以添加控件和标签，并可接收用户数据。Wizard 控件可帮助管理要显示哪个步骤及维护所收集的数据。

Wizard 控件具有线性导航和非线性导航的功能。该控件的状态管理功能允许用户在各个步骤之间前后移动，并且在显示有侧栏的情况下，还允许用户在任何时候任意选择步骤。通过使用 StepNextButtonText、StepPreviousButtonText 和 FinishCompleteButtonText 属性，可以自定义该控件的根 asp: Wizard 元素中用于导航的文本。

```
<asp: Wizard ID="Wizard1" Runat="server"  
  StepNextButtonText=" 下一步 >> "  
  StepPreviousButtonText=" << 上一步 "  
  FinishCompleteButtonText="完成! ">
```

2. AddressForm用户控件的创建

在解决方案资源管理器中右击 UserControl 文件夹，添加新项，创建 AddressForm 用户控件来实现订单接收地址和货物接收地址的填写。

(1) AddressForm 的 HTML 代码

```
<%@ControlLanguage="C#" AutoEventWireup="true" CodeFile="AddressForm.ascx.cs" Inherits="User  
Ctrl_AddressForm" %>  
<table style="width: 100%;">
```

```
<tr>  
    <td>  
        真实姓名</td>  
    <td class="style2">  
        &nbsp;<asp:TextBox ID="txtName" runat="server"></asp:TextBox>  
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"  
ControlToValidate="txtName" ErrorMessage="姓名不能为空"></asp:RequiredFieldValidator>  
    </td>  
    <td>  
        &nbsp;<br>  
    </td>  
</tr>  
<tr>  
    <td class="style1">  
        所在国家</td>  
    <td class="style2">  
        &nbsp;<asp:DropDownList ID="ddlCountry" runat="server">  
            <asp.ListItem>中华人民共和国</asp.ListItem>  
            <asp.ListItem>俄罗斯</asp.ListItem>  
            <asp.ListItem>英国</asp.ListItem>  
            <asp.ListItem>美国</asp.ListItem>  
        </asp:DropDownList>  
    </td>  
    <td>  
        &nbsp;<br>  
    </td>  
</tr>  
<tr>  
    <td class="style1">  
        省份</td>  
    <td class="style2">  
        <asp:DropDownList ID="ddlState" runat="server">  
            <asp.ListItem>吉林省</asp.ListItem>  
        </asp:DropDownList>  
    </td>  
    <td>  
        &nbsp;<br>  
    </td>  
</tr>  
<tr>  
    <td class="style1">  
        所在城市</td>  
    <td class="style2">  
        <asp:DropDownList ID="ddlCity" runat="server">  
            <asp.ListItem>长春</asp.ListItem>  
        </asp:DropDownList>  
    </td>
```

```

        <td>
            &nbsp;   </td>
        </tr>
        <tr>
            <td class="style1">
                邮编</td>
            <td class="style2">
                <asp:TextBox ID="txtZip" runat="server"></asp:TextBox>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator3" runat="server"
                ControlToValidate="txtZip" ErrorMessage="邮编格式错误"
ValidationExpression="\d{6}"></asp:RegularExpressionValidator>
            </td>
            <td>
                &nbsp;   </td>
        </tr>
        <tr>
            <td class="style1">
                接收地址</td>
            <td class="style2">
                <asp:TextBox ID="txtAddr" runat="server" Width="337px"></asp:TextBox>
            </td>
            <td>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
                ControlToValidate="txtAddr" ErrorMessage="地址不能为空"
"></asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td class="style1">
                Email</td>
            <td class="style2">
                <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server"
                ControlToValidate="txtEmail" ErrorMessage="非法的 Email 格式"

ValidationExpression="\w+([+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"></asp:RegularExpressionValidat
or>
            </td>
            <td>
                &nbsp;   </td>
        </tr>
        <tr>
            <td class="style1">
                联系电话(8 位)</td>
            <td class="style2">
                <asp:TextBox ID="txtTel" runat="server"></asp:TextBox>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"

```

```

ControlToValidate="txtTel" ErrorMessage="格式非法"
ValidationExpression="(\\d{3}\\d{3}-)?\\d{8}"></asp:RegularExpressionValidator>
</td>
<td>
    &nbsp;   </td>
</tr>
</table>

```

(2) AddressForm.ascx.cs 代码实现

```

...(略)
public partial class UserCtrl_AddressForm : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    public AddressInfo Address
    {
        get
        {
            string email=txtEmail.Text;
            string name=txtName.Text;
            string address=txtAddr.Text;
            string city=ddlCity.Text;
            string state=ddlState.Text;
            string zip=txtZip.Text;
            string country=ddlCountry.Text;
            string phone=txtTel.Text;
            //返回一个地址类实体对象
            return new AddressInfo(email,name,address,city,state,zip,country,phone);
        }
        set
        {
            //为实体类对象赋值
            if (value!=null)
            {
                txtEmail.Text = value.Email;
                txtName.Text = value.Name;
                txtAddr.Text = value.Address;
                ddlCity.Text = value.City;
                ddlState.Text = value.State;
                txtZip.Text = value.Zip;
                ddlCountry.Text = value.Country;
                txtTel.Text = value.Phone;
            }
        }
    }
}

```

3. 生成订单页CheckOut.aspx的效果图

生成订单页 CheckOut.aspx 的效果图如图 1-63 所示。



图 1-63 生成订单页 CheckOut.aspx 的效果图

4. CheckOut.aspx 的HTML代码

```
<% @PageLanguage="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true" CodeFile=
"CheckOut.aspx.cs" Inherits="CheckOut" Title="订单生成页" %>
<% @ Register src="UserCtrl/AddressForm.ascx" tagname="AddressForm" tagprefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<div style="font-size:small;text-align:left;margin-top:30px;margin-left:50px;">
<asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="0"
onfinishbuttonclick="Wizard1_FinishButtonClick"
onnextbuttonclick="Wizard1_NextButtonClick">
<WizardSteps>
<asp:WizardStep ID="WizardStep1" runat="server" Title="账单地址">
<uc1:AddressForm ID="BillingForm" runat="server" />
</asp:WizardStep>
<asp:WizardStep ID="WizardStep2" runat="server" Title="接货地址">
<asp:CheckBox ID="cbAddress" runat="server" OnCheckedChanged="cbAddress_Checked
Changed"
Text="接货地址与账单地址相同" BackColor="Red" BorderColor="Red"
AutoPostBack="True" ForeColor="white" />
<uc1:AddressForm ID="ShippingForm" runat="server" />
</asp:WizardStep>
</WizardSteps>
```



```
</asp:Wizard>
</div>
</asp:Content>
```

5. CheckOut.aspx.cs代码实现

```
public partial class CheckOut : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //获取用户购物车数据
        Profile.ShoppingCart.GetCartItems(HttpContext.Current.User.Identity.Name);
        if (!IsPostBack)
        {
            if (Profile.ShoppingCart.count==0)
            {
                this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert('
购物车中尚无选购的图书')</script>");
            }
        }
    }

    protected void Wizard1_NextButtonClick(object sender, WizardNavigationEventArgs e)
    {
    }

    //订单提交
    protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
    {
        //创建订单业务逻辑类对象
        OrderManager om = new OrderManager();
        //创建地址类对象并初始化为货物接收地址
        AddressInfo ship = ShippingForm.Address;
        //创建地址类对象并初始化为订单接收地址
        AddressInfo bill = BillingForm.Address;

        if (Profile.ShoppingCart!= null)
        {
            if (Profile.ShoppingCart.count > 0)
            {
                //将购物车数据转化为订单明细对象数组
                LineItemInfo[] lineItems = Profile.ShoppingCart.GetOrderLineItems();
                //构造订单实体类对象
                OrdersInfo order = new OrdersInfo(HttpContext.Current.User.Identity.Name, ship,
bill, Profile.ShoppingCart.Total, lineItems);
                try
                {

```

```

        //插入订单
        om.InsertOrder(order);
        //清空用户购物车商品集合
        Profile.ShoppingCart.Clear();
        //保存至 Cart 表
        Profile.ShoppingCart.SetCartItems(HttpContext.Current.User.Identity.Name);
        this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert('订单生成成功')</script>");
    }
    catch
    {
        this.ClientScript.RegisterStartupScript(this.GetType(), "",
"<script>window.alert('订单生成失败')</script>");
    }
}
else if (Profile.ShoppingCart.count == 0)
{
    this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert('
您的购物车为空不能生成订单，请您选择商品')</script>");
}
}

}

protected void cbAddress_CheckedChanged(object sender, EventArgs e)
{
    //如果订单发送地址与货物接收地址相同
    if (cbAddress.Checked)
    {
        ShippingForm.Address = BillingForm.Address;
    }
}
}

```

1.9.9 会员查看自己的订单实现

会员可以查看自己已生成的订单，当选择某一具体订单项，会显示相应的订单明细数据。

1. “我的订单”页MyOrder.aspx效果图

会员查看自己订单时打开的“我的订单”页 MyOrder.aspx 的效果图如图 1-64 所示。



图 1-64 “我的订单”页 MyOrder.aspx 的效果图

2. MyOrder.aspx页面中各个控件的属性设置及其用途

表 1-41 MyOrder.aspx 页面中各个控件的属性设置及其用途

控件类型	控件名称	主要属性设置	用途
PS.ascx	Location		站点导航
GridView	GridOrders	DataKeyNames="OrderId"	订单列表
	GridLineItems		订单明细列表

3. MyOrder.aspx 的HTML代码

```

<% @PageLanguage="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true" CodeFile=
"MyOrder.aspx.cs" Inherits="Order" Title="我的订单" %>
<% @ Register src="UserCtrl/PS.ascx" tagName="PS" tagprefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <uc1:PS ID="PS4" runat="server"/>
    <div style="margin-top:25px; margin-left:25px; font-size:small; ">
        订单明细: 请选择订单列表查看相应明细
    <asp:GridView ID="GridLineItems" runat="server" Width="95%"
        AutoGenerateColumns="False">
        <Columns>
            <asp:BoundField DataField="LineNum" HeaderText="明细编号"/>
            <asp:BoundField DataField="BookId" HeaderText="图书编号" />
            <asp:BoundField DataField="UnitPrice" HeaderText="单价"/>
        </Columns>
    </asp:GridView>
    <br />
    订单列表:
    <br />
    <asp:GridView ID="GridOrders" runat="server"
        onselectedindexchanged="GridOrders_SelectedIndexChanged"
        DataKeyNames="OrderId" EmptyDataText="当前用户没有生成订单" Width="95%"
        AutoGenerateColumns="False" CellPadding="4" ForeColor="#333333"
        GridLines="None">

```

```

<FooterStyle BackColor="#507CD1"Font-Bold="True"ForeColor="White"/>
<RowStyle BackColor="#EFF3FB"/>
<Columns>
    <asp:CommandField HeaderText="选择"ShowHeader="True"ShowSelectButton="True"/>
    <asp:BoundField DataField="OrderId" HeaderText="订单编号" />
    <asp:BoundField DataField="ShipAddr"HeaderText="账单地址"/>
    <asp:BoundField DataField="BillAddr" HeaderText="接货地址" />
    <asp:BoundField DataField="TotalPrice" HeaderText="订单总价"/>
</Columns>
<PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />
<SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />
<HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
<EditRowStyle BackColor="#2461BF" />
<AlternatingRowStyle BackColor="White" />
</asp:GridView>
</div>
</asp:Content>

```

4. MyOrder.aspx.cs代码实现

```

public partial class Order : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            //创建订单业务逻辑类对象
            OrderManager om = new OrderManager();
            //获取当前通过身份验证的用户标识
            string userId = HttpContext.Current.User.Identity.Name;
            //获取该用户订单数据，作为 GridOrders 的数据源
            GridOrders.DataSource=om.GetOrdersByUserId(userId);
            GridOrders.DataBind();
            GridOrders.Font.Size = 10;
            GridLineItems.Font.Size = 10;
        }
    }
    //选取订单列表中某行时显示与该行订单编号对应的订单明细数据
    protected void GridOrders_SelectedIndexChanged(object sender, EventArgs e)
    {
        int id = Int32.Parse(GridOrders.SelectedDataKey.Value.ToString());//获取每一行的 DataKey,
        Web 页的 DataKeyNames="OrderId"
        OrderManager om = new OrderManager();
        //依据订单编号获取订单明细数据，作为 GridLineItems 的数据源
        GridLineItems.DataSource = om.GetLineItemsByOrderId(id);
        GridLineItems.DataBind();
    }
}

```

1.9.10 任务小结

通过本任务的学习，完成订单管理功能。在实现本任务的过程中，要着重掌握 Wizard 控件的应用、数据库中存储过程与触发器的应用。

1.9.11 练习题

- 1. 自定义学生信息表及班级信息表，定义触发器，完成每当向学生信息表插入一名新学生，即在班级信息表中将“学生人数”字段值加一。
- 2. 模拟网上支付，将该步骤添加到项目设计中的 Wizard 控件中。

1.10 后台图书类别管理实现

学习目标

- 掌握存储过程的用法
- 掌握存储过程参数的定义及使用
- 建议学时：8 学时

1.10.1 任务名称：实现图书类别管理

1.10.2 任务描述

一个书店里经营的图书种类繁多，内容涵盖面也比较广泛，通常要包含十几个类别，然而这些类别并不总是一成不变的。为了快速适应图书市场的变化，灵活多样的图书类别管理可以使图书的分类管理更为有效。

图书类别管理主要包括 3 个功能。

- (1) 添加图书目录
- (2) 修改图书目录
- (3) 删除图书目录

1.10.3 任务分析

添加图书目录环节的实现方法大致为：通过在页面上部署一个 GridView 控件、三个文本框和两个按钮，其中 GridView 控件用来显示当前目录情况，文本框收集用户输入的新的目录编号、目录名称、目录描述等信息，然后通过按钮条件提交数据到业务访问层的具体方法更新数据库信息。

1.10.4 在Model类库中创建目录实体类CategoryInfo

程序开发步骤如下：

- ① 在 Model 类库中添加新项，选择类，取名为 CategoryInfo，定义所属命名空间为 namespace BookShop.Model，并标记为可序列化[Serializable]，访问修饰符设为 public。
- ② 主要程序代码：目录实体类包含 3 个内部变量，分别对应数据库中目录 Category 表中的图书信息字段。其代码如下：

```

using System;
namespace BookShop.Model
{
    [Serializable]
    public class CategoryInfo
    {
        private Int16 id;
        public Int16 Id
        {
            get { return id; }
            set { id = value; }
        }
        private string name;
        public string Name
        {
            get { return name; }
            set { name = value; }
        }
        private string description;
        public string Description
        {
            get { return description; }
            set { description = value; }
        }
    }
}

```

实体类属性的数据类型应与表中字段的数据类型相一致。

在设计与数据库中的表相对应的实体类时应重点留意表中各个字段的数据类型，实体类的属性类型应与表中的字段类型相一致。本例的 **Category** 表中包含一个 **SmallInt** 类型的 **CategoryId** 字段，SQL 数据库中的 **SmallInt** 数据类型与 C#语言中的 **Int16** 数据类型相对应，因此位于 **Model** 层的实体类 **Category** 的 **ID** 属性应该定义为 **Int16** 类型。由于数据类型不相符而造成数据读取或写入失败在初学者所犯的错误中尤为突出。

1.10.5 DAL层：目录添加数据访问类CategoryAccess类的实现

在 **DAL** 类库中添加新类目录数据访问类 **CategoryAccess**，所属命名空间为 **namespace BookShop.DAL**，访问权限为 **public**。该类的功能是对目录 **Category** 表进行数据访问，它包含 3 个主要的方法：使用 **GetCategorys** 方法能够获得保存在 **Category** 表中的所有图书目录信息；使用 **AddCategory** 方法可以添加 **Category** 表中新的目录数据，如果即将添加的目录编号与数据库中已有的目录编号存在重复，则终止当前添加目录的操作，否则完成添加目录的操作；**EditCategory** 方法用来更新指定的目录项，可以修改目录名称 **CategoryName** 和目录注释 **Desc**。

```

Public class CategoryAccess
{
    public IList<CategoryInfo> GetCategorys()

```

```

{
    Database db=new Database();
    SqlDataReader dr=db.ExecuteReader("GetCategorys");
    IList<CategoryInfo> list=new List<CategoryInfo>();

    while(dr.Read())
    {
        CategoryInfo ci=new CategoryInfo();
        ci.Id=dr.GetInt16(0);
        ci.Name=dr.GetString(1);
        ci.Description=dr.GetString(2);
        list.Add(ci);
    }
    dr.Close();
    return list;
}

public bool AddCategory(Int16 CategoryId,string name,string desc)
{
    //首先判断目录编号是否重复
    //创建参数，并赋值
    SqlParameter[] parm = {
        new SqlParameter("@CategoryId", SqlDbType.Int);
        parm[0].Value = CategoryId;
    //执行查询
    Database db = new Database();
    //验证 Category 表是否存在待插入目录数据行
    SqlDataReader rdr = db.ExecuteReader(CommandType.StoredProcedure,
        "SelectCategoryById", parm);

    if (rdr.Read())
        return false;

    //获取参数
    SqlParameter[] param ={
        new SqlParameter("@CategoryId", SqlDbType.Int),
        new SqlParameter("@CategoryName", SqlDbType.VarChar, 50),
        new SqlParameter("@CategoryDesc", SqlDbType.VarChar, 50));
    //设置参数的值

    param[0].Value = CategoryId;
    param[1].Value = name;
    param[2].Value = desc;

    db.ExecuteNonQuery(CommandType.StoredProcedure, "AddCategory", param);
    return true;
}

//编辑 Category 选中行,并更新

```

```

public void EditCategory(CategoryInfo info)
{
    Database db = new Database();
    SqlParameter[] parm = {
        new SqlParameter("@CategoryId", SqlDbType.Int),
        new SqlParameter("@CategoryName", SqlDbType.NVarChar, 50),
        new SqlParameter("@Desc", SqlDbType.NVarChar, 50)};

    parm[0].Value = info.Id;
    parm[1].Value = info.Name;
    parm[2].Value = info.Description;
    db.ExecuteNonQuery(CommandType.StoredProcedure, "UpdateCategoryById", parm);
}
}

```

1. 存储过程AddCategory的实现

在数据库中新建存储过程 AddCategory，代码如下：

```

CREATE PROCEDURE dbo.AddCategory
    @CategoryId int,
    @CategoryName nvarchar(50),
    @CategoryDesc nvarchar(50)

AS
    /* SET NOCOUNT ON */
    insert into Category (CategoryId, CategoryName, [Desc]) values (@CategoryId,@CategoryName,
    @CategoryDesc)

```

2. 存储过程UpdateCategoryById的实现

在数据库中新建存储过程 UpdateCategoryById，代码如下：

```

CREATE PROCEDURE dbo.UpdateCategoryById
    @CategoryId smallint,
    @CategoryName nvarchar(50),
    @Desc nvarchar(50)

AS
    update Category set CategoryName=@CategoryName where CategoryId=@CategoryId
    update Category set [Desc]=@Desc where CategoryId=@CategoryId

```

3. 存储过程SelectCategoryById

在数据库中新建存储过程 SelectCategoryById，代码如下：

```

CREATE PROCEDURE dbo.SelectCategoryById
    @CategoryId int

AS
    select * from Category where CategoryId=@CategoryId

```


4. 存储过程GetCategorys

在数据库中新建存储过程 GetCategorys，代码如下：

```
CREATE PROCEDURE dbo.GetCategorys
/*
(
    @parameter1 int = 5,
    @parameter2 datatype OUTPUT
)
*/
AS
/* SET NOCOUNT ON */
select CategoryId, CategoryName, [desc] from Category
```

5. 相关技能知识点

表中字段与数据库查询语句同名：当用户自定义的表中含有与数据库查询语句同名的字段时，应该在 SQL 语句中将该字段写成[×××]。如 select CategoryId, CategoryName, [desc] from Category，其中 desc 字段在表中的含义为目录详情，但它与数据库查询语句中的关键字“Desc”重名，因此在使用该字段时要求加上方括号“[]”，否则查询出错。

1.10.6 BLL层： 目录添加管理业务逻辑类CategoryManager类的实现

在 BLL 类库中添加新类，命名为 CategoryManager，定义所属命名空间为 namespace BookShop.BLL，标记为可序列化[Serializable]，访问权限为 public。该类负责添加新的图书目录到数据库中，对数据库中 Category 表的操作，该类会调用目录数据访问类 CategoryAccess 来完成。

```
Public class CategoryManager
{
    private static readonly CategoryAccess ca = new CategoryAccess();
    public IList<CategoryInfo> GetCategoryInfos()
    {
        return ca.GetCategorys();
    }
    //添加目录
    public bool AddCategory(Int16 CategoryId, string name, string desc)
    {
        return ca.AddCategory(CategoryId, name, desc);
    }
}
```

1.10.7 目录添加管理表示层代码实现

图书目录添加管理在表示层包含目录管理页面 AddCategory.aspx 及其代码文件 AddCategory.aspx.cs。

1. 后台管理母版页的设计

在 admin 文件夹下添加母版页，其 HTML 代码如下：

```

<% @ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
Inherits="admin_MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>无标题页</title>
    <script type="text/javascript">
window.onload=window.onresize=function(){
if(document.getElementById("left").clientHeight<document.getElementById("right").clientHeight){
document.getElementById("left").style.height=document.getElementById("right").offsetHeight+"px";
}
else{
document.getElementById("right").style.height=document.getElementById("left").offsetHeight+"px";
}
}
</script>
</head>
<body style="background-image:url('/images/bgbody.gif')">
    <form id="form1" runat="server">
    <div id="total"
        style="background-color: #FFFFFF;
        border: 1px solid Gray;
        margin: 3px auto 3px auto;
        width: 900px;
        height: 675px">
        <div id="header"
            style="background-image: url('/images/title1.gif');
            border-bottom-style: dotted;
            border-bottom-width: 1px;
            border-bottom-color: Gray;
            width: 900px;
            height: 120px">

        </div>

        <div id="menu"
            style="line-height: 30px;
            background-image:url('/images/bgg0.jpg');
            text-align: left;
            border-bottom-style: dotted;
            border-bottom-width: 1px;
            border-bottom-color: Gray;
            padding-left: 45px;
            width: 855px;

```

```

height: 30px;
overflow: hidden;">
<!--menu 层的 系统时间 显示标签 和 超链接-->
    <span style=" margin-left: 10px"></span>
    <asp:Label ID="lbltime" runat="server" Text=""></asp:Label>

    <span style=" margin-left: 100px"></span>
    <a href="CategoryManager.aspx" style=" color:Blue">目录管理</a>
    <a href="BookManager.aspx" style=" color:Blue">图书管理</a>

<!-->
</div>
<div id="left"
    style=" background-image:url('../images/bgg1.jpg');
    font-size: small;

    border-right-style: dotted;
    border-right-width: 1px;
    border-right-color: Gray;
    padding: 2px;
    width: 175px;
    height: 500px; <%--500px--%>
    float: left
">
<!--left 层的超链接-->
    <p align="center" style=" font-size: large; padding-top: 100px">
        <a href="CategoryManager.aspx" style=" color:Blue">目录管理</a>
    </p>
    <p align="center" style=" font-size: large; padding-top: 20px">
        <a href="BookManager.aspx" style=" color:Blue">图书管理</a>
    </p>
<!-->
</div>

<div id="right"
style="width: 720px;
    height: 500px;
    float: left;
    background-image:url('../images/bgg2.jpg');">
    <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

    </asp:ContentPlaceHolder>

</div>
<div id="footer"
style="font-size: small;
    text-align: center;

```

```

background-image: url('/images/xiajy.jpg');
border-top-style: dotted;
border-top-width: 1px;
border-top-color: Gray;
padding-top: 2px;
width: 900px;
height: 80px"><%--80px--%>
<br /> 版权所有：长春职业技术学院软件技术专业
<br /> 学院地址：长春市卫星路 3278 号[130033]
        招生热线：0431-84602444 84602555 84602666
<br />
        请用 IE5.0 以上版本 1024×768 以上分辨率浏览
        就业电话：0431-84602777 84602750

</div>
</div>
</form>
</body>
</html>

```

后台管理首页效果图如图 1-65 所示。



图 1-65 后台管理首页效果图

2. 后台管理页default.aspx

在网站根目录添加一个文件夹，命名为 admin。

在文件夹 admin 中新建一个 Web 窗体，命名为 default.aspx，并选择 MasterPage.master 为母版页。

3. 目录添加页AddCategory.aspx的设计

在 admin 中新建一个 Web 窗体，命名为 AddCategory.aspx，并选择 MasterPage.master 为母版页。

```

<% @PageLanguage="C#" MasterPageFile="~/admin/MasterPage.master" AutoEventWireup="true"
CodeFile="AddCategory.aspx.cs" Inherits="admin_AddCategory" Title="无标题页" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">

<div id="divShowCategory" align="center">
<br />
<label>添加目录管理页面</label>

```

```

        <br />
<asp:LabelID="lblmsg"runat="server"Text=""ForeColor="Red"Font-Bold="True"></asp:Label>
        <br />
        <asp:GridView ID="gvCategorys" runat="server"></asp:GridView>
    </div>

    <div id="divCategoryName">
        <table align="center">
            <tr>
                <td>目录编号:</td>
                <td><asp:TextBox ID="textboxCategoryId" runat="server" Height="22px"
Width="430px"
                    MaxLength="30" Wrap="False"></asp:TextBox></td>
            </tr>
            <tr>
                <td>目录名称:</td>
                <td><asp:TextBox ID="textboxCategoryName" runat="server" Height="22px"
Width="430px"
                    MaxLength="30" Wrap="False"></asp:TextBox></td>
            </tr>
            <tr>
                <td>目录描述:</td>
                <td><asp:TextBoxID="textboxCategoryDesc"runat="server"Width="430px"Rows="5"
                    TextMode="MultiLine"></asp:TextBox></td>
            </tr>
        </table>
    </div>
    <div id="divBtnv "style="text-align: center">
        <asp:Button ID="btnAddOk" runat="server" Text="添加" Width="85px"
            onclick="btnAddOk_Click" />
        <asp:Button ID="btnAddCancel" runat="server" Text="返回" Width="85px"
            onclick="btnAddCancel_Click" />
    </div>
</asp:Content>

```

AddCategory.aspx 页面中各个控件的属性设置及其用途如表 1-42 所示。

表 1-42 AddCategory.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
Label	lblmsg	Text 属性在执行完添加或删除目录操作后由系统自动生成	显示添加目录操作成功与否
asp: TextBox	textboxCategoryId	MaxLength 目录编号的最大长度	设定新加入的目录编号
asp: TextBox	textboxCategoryName	MaxLength 目录名称的最大长度	设定新加入的目录名称

续表

控件类型	控件名称	主要属性设置	用途
asp: TextBox	txtboxCategoryDesc	MaxLength 目录描述的最大长度	设定新加入的目录描述
ImageButton	btnDelete	CommandName 属性设置为"Del" CommandArgument 属性设置为'<%# Eval("BookId") %>'	执行从购物车中移除图书的操作
TextBox	TxtQty	Text 属性设置为'<%# Eval("Quantity") %>'	显示购买图书的数量
Button	btnAddOk	onclick="btnAddOk_Click"	执行添加目录的操作
	btnAddCancel	onclick="btnAddCancel_Click"	返回后台管理页面 Default.aspx

目录添加页效果图如图 1-66 所示。



图 1-66 目录添加页效果图

4. 代码文件AddCategory.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    this.Bind();
}
//填充页面 Gridview 控件的数据源：目录
public void Bind()
{
    CategoryManager cm = new CategoryManager();
    gvCategories.DataSource = cm.GetCategoryInfos();
    gvCategories.DataBind();
}

//当用户单击“返回”按钮时将浏览器显示的页面跳转到 CategoryManager.aspx
protected void btnAddCancel_Click(object sender, EventArgs e)
```

```

{
    Response.Redirect("CategoryManager.aspx");
}

//添加目录信息到 Category 表
protected void btnAddOk_Click(object sender, EventArgs e)
{
    string id = txtboxCategoryId.Text.Trim();
    string name = txtboxCategoryName.Text.Trim();
    string desc = txtboxCategoryDesc.Text.Trim();

    //添加之前保证等待插入的数据项不能包含空串
    if (id != "" && name != "" && desc != "")
    {
        Int16 CategoryId = Int16.Parse(id);
        CategoryManager cm = new CategoryManager();

        //调用 DAL 包的 AddCategory(CategoryId, name, desc)
        bool b=cm.AddCategory(CategoryId, name, desc);
        //添加成功, lblmsg 显示成功
        if (b)
        {
            lblmsg.Text = "添加目录: " + name + " 成功";
            this.Bind();
            txtboxCategoryId.Text = "";
            txtboxCategoryName.Text = "";
            txtboxCategoryDesc.Text = "";
        }
        else
        {
            lblmsg.Text = "已经存在该目录, 添加: "+name +"失败";
        }
    }
    else
    {
        this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>alert(' 输入 数据 不 完 整!')</script>");
    }
}

```

5. 相关技能知识点

当业务逻辑需要在某种条件下弹出对话框时，一般的做法是使用 `this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>alert('输入数据不完整!')</script>");` 语句，该语句为客户端脚本语句，使用它不会造成页面布局异常。

1.10.8 目录管理功能实现

1. DAL层: CategoryAccess类完善目录管理功能

在 DAL 类库下添加新项, 选择类, 命名为 CategoryAccess, 加入以下代码:

```
namespace BookShop.DAL
{
    public class CategoryAccess
    {
        public IList<CategoryInfo> GetCategorys()
        {
            Database db=new Database();
            SqlDataReader dr=db.ExecuteReader("GetCategorys");
            IList<CategoryInfo> list=new List<CategoryInfo>();

            while(dr.Read())
            {
                CategoryInfo ci=new CategoryInfo();
                ci.Id=dr.GetInt16(0);
                ci.Name=dr.GetString(1);
                ci.Description=dr.GetString(2);
                list.Add(ci);
            }
            dr.Close();
            return list;
        }

        public bool AddCategory(Int16 CategoryId, string name, string desc)
        {
            //首先判断目录编号是否重复
            //创建参数, 并赋值
            SqlParameter[] parm = {
                new SqlParameter("@CategoryId", SqlDbType.Int)};
            parm[0].Value = CategoryId;
            //执行查询
            Database db = new Database();
            SqlDataReader rdr=db.ExecuteReader(CommandType.StoredProcedure, "SelectCategoryById", parm);

            if (rdr.Read())
                return false;

            //获取参数
            SqlParameter[] param ={
                new SqlParameter("@CategoryId",SqlDbType.Int),
                new SqlParameter("@CategoryName",SqlDbType.VarChar, 50),
                new SqlParameter("@CategoryDesc",SqlDbType.VarChar, 50)};

            //设置参数的值
```



```

param[0].Value = CategoryId;
param[1].Value = name;
param[2].Value = desc;

db.ExecuteNonQuery(CommandType.StoredProcedure,"AddCategory",param);
return true;
    }
    public bool DelCategory(int CategoryId)
    {
//判断 BookBrief 表是否存在 CategoryId 的记录
SqlParameter[] parm = {
        new SqlParameter("@CategoryId", SqlDbType.Int)};

parm[0].Value = CategoryId;

//执行查询
Database db = new Database();
SqlDataReader dr = db.ExecuteReader(CommandType.StoredProcedure,
        "SelectBookBriefById",parm);

//在 BookBrief 中找到用户输入的 CategoryId 的记录
if (dr.Read())
{
        dr.Close();
        //返回 false，删除失败
        return false;
}
dr.Close();

//执行删除操作
SqlParameter[] parm2 = {
        new SqlParameter("@CategoryId", SqlDbType.Int)};

parm2[0].Value = CategoryId;

db.ExecuteNonQuery(CommandType.StoredProcedure,
        "DelCategory", parm2);

return true;
    }
    //编辑 Category 选中行，并更新
    public void EditCategory(CategoryInfo info)
    {
Database db = new Database();
SqlParameter[] parm = {
        new SqlParameter("@CategoryId", SqlDbType.Int),
        new SqlParameter("@CategoryName", SqlDbType.NVarChar, 50),

```

```

        new SqlParameter("@Desc", SqlDbType.NVarChar, 50));

    parm[0].Value = info.Id;
    parm[1].Value = info.Name;
    parm[2].Value = info.Description;
    db.ExecuteNonQuery(CommandType.StoredProcedure, "UpdateCategoryById", parm);
    }
}
}

```

2. 存储过程SelectBookBriefById

在数据库中新建存储过程 SelectBookBriefById，代码如下：

```

ALTER PROCEDURE dbo.SelectBookBriefById
    @CategoryId int
AS
    select * from BookBrief where CategoryId=@CategoryId

```

3. 存储过程DelCategory

在数据库中新建存储过程 DelCategory，代码如下：

```

ALTER PROCEDURE dbo.DelCategory
    @CategoryId int
AS
    delete from Category where CategoryId=@CategoryId

```

1.10.9 BLL层：CategoryManager类完善目录删除功能

```

namespace BookShop.BLL
{
    //[Serializable]添加在 CategoryInfo.cs 类声明之前
    public class CategoryManager
    {
        private static readonly CategoryAccess ca = new CategoryAccess();
        public IList<CategoryInfo> GetCategoryInfos()
        {
            return ca.GetCategories();
        }
        //添加目录
        public bool AddCategory(Int16 CategoryId, string name, string desc)
        {
            return ca.AddCategory(CategoryId, name, desc);
        }
        //删除目录
        public bool DelCategory(int CategoryId)
        {
            return ca.DelCategory(CategoryId);
        }
    }
}

```

```

        //修改目录
//根据 CategoryId 更新数据
        public void UpdateCategoryById(CategoryInfo info)
        {
            ca.EditCategory(info);
        }
    }
}

```

1.10.10 目录删除表示层设计

在网站文件夹 admin 下添加新项，选择 web 窗体，命名为 DelCategory，即删除目录表示层。在<asp: Content></asp: Content>标签对之间添加如下代码：

```

<div id="ShowCategorys"align="center">
    <br />
<label>删除目录管理页面</label>
    <br />
    <asp:LabelID="lblmsg"runat="server"Font-Bold="True"ForeColor="Red"></asp:Label>
    <br />
<asp:GridView ID="gvCategorys" runat="server"></asp:GridView>
</div>
<div id="Del">
<table align="center">
    <tr>
<td>
        <label>选取目录:</label>
</td>
<td>
        <asp:DropDownList ID="DropDownList1"runat="server"
            DataTextField="Name"DataValueField="Id"></asp:DropDownList>
</td>
<td>
<asp:Button ID="btnDel"runat="server"Text="删除目录"onclick="btnDel_Click"
            UseSubmitBehavior="False"/>
</td>
<td>
<asp:Button ID="btnAddCancel"runat="server"Text="返回"Width="85px"
            onclick="btnDelCancel_Click"/>
</td>
</tr>
</table>
</div>

```

DelCategory.aspx 页面中各个控件的属性设置及其用途如表 1-43 所示。

表 1-43 DelCategory.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
Asp:Label	lblmsg	Text 属性在执行完删除目录操作后由系统自动生成	显示删除目录操作成功与否
asp: GridView	gvCategorys	由后台设置数据源	显示当前目录内容
asp: DropDownList	DropDownList1	由后台填充目录项	选择目录项进而删除该目录项
asp: Button	btnDel	Text 属性为删除目录	提交删除操作
asp: Button	btnDelCancel	Text 属性为返回	退出当前界面

目录删除页效果图如图 1-67 所示。



图 1-67 目录删除页效果图

代码文件 DelCategory.aspx.cs 的内容如下：

```
using BookShop.BLL;
public partial class admin_DelCategory : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if(!IsPostBack)
        {
            this.Bind();
        }
    }
    //填充页面 Gridview 控件的数据源:目录
    public void Bind()
    {
        CategoryManager cm = new CategoryManager();
        gvCategorys.DataSource = cm.GetCategoryInfos();
        gvCategorys.DataBind();

        //填充下拉列表框
```

```

        CategoryManager cm1 = new CategoryManager();
        DropDownList1.DataSource = cm1.GetCategoryInfos();
        DropDownList1.DataBind();
    }
    protected void btnDel_Click(object sender, EventArgs e)
    {
        int CategoryId = int.Parse(DropDownList1.SelectedValue);
        CategoryManager cm = new CategoryManager();
        bool b = cm.DelCategory(CategoryId);

        if (b)
        {
            lblmsg.Text = "删除成功!";
            this.Bind();
        }
        else
        {
            lblmsg.Text = "抱歉，目录中可能存在图书，删除失败!";
        }
    }
    protected void btnAddCancel_Click(object sender, EventArgs e)
    {
        Response.Redirect("CategoryManager.aspx");
    }
}

```

1.10.11 目录修改表示层EditCategory.aspx代码实现

```

<% @PageLanguage="C#" MasterPageFile="~/admin/MasterPage.master" AutoEventWireup="true"
CodeFile="EditCategory.aspx.cs" Inherits="admin_EditCategory" Title="无标题页" %>
<asp:ContentID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <table style="text-align:center; width:650px">
        <tr>
            <td colspan="2">
                <asp:GridView ID="GridView1" runat="server"
                    AutoGenerateSelectButton="True"
                    onselectedindexchanged="GridView1_SelectedIndexChanged"
AllowPaging="True"
                    onpageindexchanging="GridView1_PageIndexChanging">
                </asp:GridView>
            </td>
        </tr>
        <tr>
            <td style="text-align:right; width: 302px;">CategoryId</td>
            <td style="text-align:left"><asp:Label ID="lblCategoryId" runat="server"
Text=""></asp:Label></td>

```

```

        </tr>
        <tr>
            <td style="text-align:right; width: 302px;">CategoryName</td>
            <td style="text-align:left"><asp:TextBoxID="txtCategoryName"runat="server"></asp:TextBox></td>
        </tr>
        <tr>
            <td style="text-align:right; width: 302px;">CategoryDesc</td>
            <td style="text-align:left"><asp:TextBoxID="txtCategoryDesc"runat="server"></asp:TextBox></td>
        </tr>

        <tr>
            <td style="text-align:right; width: 302px;">
                <asp:ButtonID="btnOK"runat="server"Text="修改"onclick="btnOK_Click" />
            </td>
            <td style="text-align:left">
                <asp:ButtonID="btnCencal"runat="server"Text="返回"onclick="btnCencal_Click" />
            </td>
        </tr>
    </table>
</asp:Content>

```

目录修改页 EditCategory 页面中各个控件的属性设置及其用途如表 1-44 所示。

表 1-44 目录修改页 EditCategory 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
asp: GridView	GridView1	AutoGenerateSelectButton="True"	显示“选择”按钮
		onselectedindexchanged="GridView1_SelectedIndexChanged"	当单击“选择”按钮时触发 GridView1_SelectedIndex Changed 事件
		AllowPaging="True"	启用分页
		onpageindexchanging="GridView1_PageIndexChanging"	当单击分页索引时触发 GridView1_PageIndexChanging 事件
asp: Label	lblCategoryId		Text 属性为返回选择目录行的目录 ID
asp: TextBox	txtCategoryName		Text 属性为返回选择或新输入的目录行的目录名称
asp: TextBox	txtCategoryDesc		Text 属性为返回选择或新输入的目录行的目录注释
asp: Button	btnOK	onclick="btnOK_Click"	当单击 btnOK 按钮时触发 btnOK_Click 事件
asp: Button	btnCancel	onclick=" btnCancel _Click"	当单击“btnCancel”按钮时触发 btnCancel_Click 事件

目录修改页效果图如图 1-68 所示。



图 1-68 目录修改页效果图

虽然常见的页面布局方式为 DIV+CSS，但是合理运用表格标签<table></table>可以快速组织页面布局。

1. 代码文件EditCategory.aspx.cs的实现

```
public partial class admin_EditCategory : System.Web.UI.Page
{
    private readonly CategoryManager cm = new CategoryManager();
    protected void Page_Load(object sender, EventArgs e)
    {
        this.myDataBind();
    }
    protected void myDataBind()
    {
        GridView1.DataSource = cm.GetCategoryInfos();
        GridView1.DataKeyNames=new string[]{"id"};
        GridView1.DataBind();
    }

    protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
    {
        lblCategoryId.Text = GridView1.SelectedRow.Cells[1].Text;
        txtCategoryName.Text = GridView1.SelectedRow.Cells[2].Text;
        txtCategoryDesc.Text = GridView1.SelectedRow.Cells[3].Text;
    }
    protected void btnOK_Click(object sender, EventArgs e)
    {
        string name = txtCategoryName.Text.Trim();
        string desc = txtCategoryDesc.Text.Trim();
        if (name != "" && desc != "")
        {
            CategoryInfo info = new CategoryInfo();
            info.Id = Int16.Parse(GridView1.SelectedRow.Cells[1].Text);
            info.Name = txtCategoryName.Text.Trim();
            info.Description = txtCategoryDesc.Text.Trim();
            cm.UpdateCategoryById(info);
        }
    }
}
```

```

    }
    else
    {
        this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert('请先
单击【选择】!')</script>");
    }

    this.myDataBind();
}
protected void btnCancel_Click(object sender, EventArgs e)
{
    Response.Redirect("CategoryManager.aspx");
}

protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    GridView1.PageIndex = e.NewPageIndex;
    this.myDataBind();
}
}

```

2. 相关技能知识点

当使用 GridView 控件进行数据写入或选择操作时需要设定 GridView 控件的 DataKeyNames 属性，使用 DataKeyNames 属性指定表示数据源主键的字段。

当在 GridView 控件中选中某行时，请使用 SelectedRow 属性检索表示该行的 GridViewRow 对象。例如，打算获取选中行的第一列字段的数据时，其用法为 GridView1.SelectedRow.Cells[1].Text；值得注意的是，该Cells[xx]的下标是从 1 开始的，且其值不能超过数据库查询语句Select检索的数据列数。

1.10.12 目录管理页CategoryManager.aspx

在 admin 文件夹下添加新项 Web 窗体，命名为 CategoryManager。添加以下代码：

```

<%@PageLanguage="C#"MasterPageFile=~\admin\MasterPage.master"AutoEventWireup="true"
CodeFile="CategoryManager.aspx.cs" Inherits="admin_CategoryManager" Title="无标题页" %>
<asp:ContentID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <div align="center">
        <asp:GridViewID="gvCategories"runat="server"style="text-align:center"></asp:GridView>
        <asp:ButtonID="btnAdd"runat="server" Text="添加目录" onclick="btnAdd_Click" />
        <asp:Button ID="btnDel" runat="server" Text="删除目录" onclick="btnDel_Click"
style="text-align: center" />
        <asp:ButtonID="btnEdit"runat="server"Text="修改目录"onclick="btnEdit_Click"/>
        <asp:Button ID="btnCancel" runat="server" Text="返回" Width="85px"
onclick="btnCancel_Click" />

    </div>
</asp:Content>

```


CategoryManager.aspx 页面中各个控件的属性设置及其用途如表 1-45 所示。

表 1-45 CategoryManager.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
asp: GridView	gvCategories		显示目录表记录
asp: Button	btnAdd	onclick="btnAdd_Click"	单击按钮时触发 btnAdd_Click 事件
asp: Button	btnDel	onclick="btnDel_Click"	单击按钮时触发 btnDel_Click 事件
asp: Button	btnEdit	onclick="btnEdit_Click"	单击按钮时触发 btnEdit_Click 事件
asp: Button	btnAddCancel	onclick="btnCancel_Click"	单击按钮时触发 btnCancel_Click 事件

目录管理页效果图如图 1-69 所示。



图 1-69 目录管理页效果图

目录管理页面 CategoryManager.aspx.cs 代码如下：

```
public partial class admin_CategoryManager : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        this.Bind();
    }

    public void Bind()
    {
        CategoryManager cm=new CategoryManager();
        gvCategories.DataSource = cm.GetCategoryInfos();
        gvCategories.DataBind();
    }

    protected void btnAdd_Click(object sender, EventArgs e)
    {

```

```

        Response.Redirect("AddCategory.aspx");
    }
    protected void btnDel_Click(object sender, EventArgs e)
    {
        Response.Redirect("DelCategory.aspx");
    }
    protected void btnCancel_Click(object sender, EventArgs e)
    {
        Response.Redirect("Default.aspx");
    }
    protected void btnEdit_Click(object sender, EventArgs e)
    {
        Response.Redirect("EditCategory.aspx");
    }
}

```

1.10.13 任务小结

在实现图书类别管理的任务中遵循三层标准架构，主要实现了目录管理的数据访问类、目录管理的业务逻辑类、目录管理的相关页面及代码等，在对数据访问时使用了存储过程，在存取目录管理数据时应用了可按照索引单独访问的对象的非泛型集合 `IList<>` 接口，在表示层主要使用了 `GridView` 服务器控件。通过对以上各技术的合理运用，较好地解决了在任务分析过程中所提出的 3 个主要问题，即添加目录、删除目录、修改目录。

本任务在解决添加目录环节中定义了 `AddCategory` 方法，该方法的解题思路是按照新添加目录的目录编号 `CategoryId` 检索数据库，查看是否存在相同目录编号，如果存在则给出提示：目录编号重复，请求重新添加。而这个检索目录编号是否重复的操作则是通过调用一个带参数的存储过程查询数据库得来的。实际上，在添加目录管理页面上既提供了用于接收新目录的相关文本框，又提供了当前的所有目录，而此时这个所有目录的提示是通过使用一个 `GridView` 来呈现的，其中用到了一个返回 `List<Category>` 类型数据的方法 `getCategories`。

在前面章节已经学习了泛型集合类 `Dictionary<Tkey, TValue>` 的用法，该泛型集合类的一个显著特点就是可以对其内部存储的键值对进行快速的检索，为此而消耗的系统资源却又相对较少。因此完全可以利用泛型集合类 `Dictionary<Tkey, TValue>` 的这个特性，改写添加目录 `AddCategory` 方法内部的实现逻辑。关键之处在于需要在位于 DAL 层的 `CategoryAccess` 类中添加一个泛型集合类的私有成员，并且要求该成员是 `static` 类型的，在 `AddCategory` 方法中检索该私有成员中是否含有新添加目录的 `CategoryId` 键值，同时还要修改获取所有目录的方法 `GetCategories`。

例如：

```

Private static Dictionary<int, Category> dic=new Dictionary<int, Category>();
...
public IList<CategoryInfo> GetCategories()
{
    Database db=new Database();
    SqlDataReader dr=db.ExecuteReader("GetCategories");
}

```

```

        IList<CategoryInfo> list=new List<CategoryInfo>();
        Dic.Clear();
        while(dr.Read())
        {
            CategoryInfo ci=new CategoryInfo();
            ci.Id=dr.GetInt16(0);
            ci.Name=dr.GetString(1);
            ci.Description=dr.GetString(2);
            dic.Add(ci.CategoryId, ci);
        }
        dr.Close();
        foreach(Category c in dic.Values)
        {
            List.Add(c);
        }

        return list;
    }

    ...
    Public bool AddCategory(Int16 CategoryId, string name, string desc)
    {
        If(dic.ContainsKey(CategoryId))
        {
            Return false;
        }
        Else
        {
            ...
        }
    }

```

1.10.14 练习题

1. 完善程序：存储过程的正确调用。

```

public void EditCategory(CategoryInfo info)
{
    Database db = new Database();
    SqlParameter[] parm = {
        new SqlParameter("@CategoryId", SqlDbType.Int),
        new SqlParameter("@CategoryName", SqlDbType.NVarChar, 50),
        new SqlParameter("_____", SqlDbType.NVarChar, 50)};
    //比照数据库中的存储过程,写全参数
    parm[0].Value = info.Id;
    parm[1].Value = info.Name;
    parm[2].Value = info.Description;
    db.ExecuteNonQuery(_____//设定 CommandType 类型
    , "UpdateCategoryById", parm);
}

```

2. 已知 Category 表的结构如表 1-46 所示，请编写用于查询 Category 表中全部数据的存储过程。

表 1-46 Category 表结构

字段名称	字段类型	字段含义
CategoryId	SmallInt	目录编号
CategoryName	Nvarchar(50)	目录名称
Desc	Nvarchar(50)	目录注释

1.11 后台图书信息管理

学习目标

- 掌握更新多个具有外键关联的表的方法
- 存储过程的应用
- 存储过程参数的定义及使用
- 建议学时：8 学时

1.11.1 任务名称： 后台图书信息管理

1.11.2 任务描述

用户通过网上购书系统所看到的各种图书信息最终来源于用来保存图书信息的数据库，对于数据库信息的更新通常情况下是由程序自动完成的，而这个程序一般情况下被设计成后台管理程序的一部分。对于图书信息管理功能模块而言，主要应该提供 3 个功能：

- (1) 添加图书
- (2) 修改概要
- (3) 删除图书

1.11.3 任务分析

添加图书时需要后台操作人员提供图书编号、图书名称、图书售价、图书数量、所属目录类别、是否为热门推荐、是否为经典推荐等信息，这些内容可以通过文本框和下拉列表框来收集。在更新开始时需要注意的是，要先更新 BookBrief 表，然后再更新 Item 表，这是因为两个表之间存在外键关系，Item 表中的 BookId 来源于 BookBrief 表的 BookId。

修改概要环节则较为简单，根据输入的图书编号，修改图书的热点和经典类型即可。其中涉及一个文本框和两个下拉列表框。

1.11.4 Model层：图书详情BookBriefInfo类实现

程序开发步骤如下。

① 在 Model 类库下添加新项，选择类，取名为 BookBriefInfo，定义所属命名空间为 namespace BookShop.Model，并标记为可序列化[Serializable]，访问修饰符设为 public。

② 主要程序代码：图书详情包括 8 个属性和两个构造方法，后者分别为无参构造方法和有参构造方法。

```
namespace BookShop.Model
{

    /// <summary>
    /// Business entity used to model a product
    /// </summary>
    [Serializable]
    public class BookBriefInfo {

        // Internal member variables

        private string bookId;
        public string BookId
        {
            get { return bookId; }
            set { bookId = value; }
        }

        private string bookName;
        public string BookName
        {
            get { return bookName; }
            set { bookName = value; }
        }

        private string bookDesc;
        public string BookDesc
        {
            get { return bookDesc; }
            set { bookDesc = value; }
        }

        private string bookImage;
        public string BookImage
        {
            get { return bookImage; }
            set { bookImage = value; }
        }

        private string categoryId;
        public string CategoryId
        {
            get { return categoryId; }
            set { categoryId = value; }
        }
    }
}
```

```

    }

    private DateTime addTime;
    public DateTime AddTime
    {
        get { return addTime; }
        set { addTime = value; }
    }

    private bool isHeadLine;
    public bool IsHeadLine
    {
        get { return isHeadLine; }
        set { isHeadLine = value; }
    }

    private bool isClassic;
    public bool IsClassic
    {
        get { return isClassic; }
        set { isClassic = value; }
    }
    /// <summary>
    /// Default constructor
    /// </summary>
    public BookBriefInfo() { }
    /// <summary>
    /// Constructor with specified initial values
    /// </summary>
    public BookBriefInfo(string id, string name, string description, string image, string
categoryId,DateTime addtime,bool headline,bool classic) {
        this.bookId = id;
        this.bookName = name;
        this.bookDesc = description;
        this.bookImage = image;
        this.categoryId = categoryId;
        this.addTime = addtime;
        this.isHeadLine = headline;
        this.isClassic = classic;
    }
}
}
}

```

1.11.5 DAL层：图书详情访问类BookBriefAccess类的实现

在 DAL 类库下,添加新类图书详情访问类 BookBriefAccess 类所属命名空间为 namespace BookShop.DAL, 访问权限为 public。该类的功能为对图书详情表 BookBrief 表进行访问, 它

包括两个主要方法：使用 AddBookBrief 添加新书的各种信息，使用 UpdateBookBriefByBookId 根据图书编号修改图书的热点和经典类型。

1. 图书详情访问类BookBriefAccess类两个主要方法的实现

```
namespace BookShop.DAL
{
    public class BookBriefAccess
    {
        //更新
        public void AddBookBrief(BookBriefInfo bbf)
        {
            SqlParameter[] parm = {
                new SqlParameter("@BookId", SqlDbType.VarChar, 10),
                new SqlParameter("@CategoryId", SqlDbType.Int),
                new SqlParameter("@IsHeadLine", SqlDbType.Bit),
                new SqlParameter("@IsClassic", SqlDbType.Bit),
                new SqlParameter("@BookName", SqlDbType.VarChar, 80)};

            parm[0].Value = bbf.BookId;
            parm[1].Value = bbf.CategoryId;
            parm[2].Value = bbf.IsHeadLine;
            parm[3].Value = bbf.IsClassic;
            parm[4].Value = bbf.BookName;

            Database db=new Database();
            SqlDataReader dr = db.ExecuteReader(
                CommandType.StoredProcedure,
                "SelectBookBriefByBookId", parm);

            //有库存，则更新热点、经典、推荐类型及书名
            if (dr.Read())
            {
                db.ExecuteNonQuery(CommandType.StoredProcedure, "UpdateBookBriefIs", parm);
            }
            else
            {
                db.ExecuteNonQuery(CommandType.StoredProcedure, "InsertBookBrief", parm);
            }
        }

        //执行图书概要修改操作
        public void UpdateBookBriefByBookId(BookBriefInfo bbf)
        {
            Database db = new Database();
            SqlParameter[] parm = {
                new SqlParameter("@BookId", SqlDbType.VarChar, 10),
                new SqlParameter("@IsHeadLine", SqlDbType.Bit),
```

```

        new SqlParameter("@IsClassic", SqlDbType.Bit));

        parm[0].Value = bbf.BookId;
        parm[1].Value = bbf.IsHeadLine;
        parm[2].Value = bbf.IsClassic;

        db.ExecuteNonQuery(CommandType.StoredProcedure,
            "UpdateBookBriefByBookId",
            parm);
    }
    //删除指定 bookid 的图书
    public void DelBookBriefByBookId(int id)
    {
        Database db = new Database();
        SqlParameter[] parm = {
            new SqlParameter("@BookId", SqlDbType.Int));
        parm[0].Value = id;
        db.ExecuteNonQuery(CommandType.StoredProcedure, "DelBookBriefByBookId", parm);
    }
    //获取全部图书
    public SqlDataReader getDataReader()
    {
        Database db = new Database();
        return db.ExecuteReader(CommandType.StoredProcedure, "getDataReader", null);
    }
}
}

```

2. 创建添加存储过程

在添加图书详情过程中涉及调研两个存储过程，分别为 UpdateBookBriefIs 和 InsertBookBrief。需要在 SQL2005 下 CVITBOOKSHOP 数据库的可编程性文件夹下添加这两个存储过程，源代码如下。

UpdateBookBriefIs 存储过程代码为：

```

CREATE PROCEDURE dbo.UpdateBookBriefIs
    @BookId varchar(10),
    @CategoryId smallint,
    @IsHeadLine Bit,
    @IsClassic Bit,
    @BookName varchar(80)
AS
    updateBookBriefsetIsHeadLine=@IsHeadLinewhereBookId=@BookIdandCategoryId=@CategoryId
    updateBookBriefset IsClassic=@IsClassic where BookId=@BookId and CategoryId=@CategoryId
    updateBookBriefsetBookName=@BookNamewhereBookId=@BookIdandCategoryId=@CategoryId

```

InsertBookBrief 存储过程代码为：


```

CREATE PROCEDURE InsertBookBrief
@BookId varchar(10),
@CategoryId smallint,
@IsHeadLine Bit,
@IsClassic Bit,
@BookName varchar(80)
AS
insert into BookBrief
(BookId, CategoryId, IsHeadLine, IsClassic, BookName)
values (@BookId, @CategoryId, @IsHeadLine, @IsClassic, @BookName)

```

3. 创建修改存储过程

在修改图书热点和经典类型模块中涉及调用一个存储过程 UpdateBookBriefByBookId, 同样需要在 SQL2005 的 CVITBOOKSHOP 数据库的可编程性文件夹下新建存储过程 UpdateBookBriefByBookId, 用来执行图书概要修改操作。

UpdateBookBriefByBookId 存储过程代码如下:

```

CREATE PROCEDURE dbo.UpdateBookBriefByBookId
@BookId varchar(10),
@IsHeadLine Bit,
@IsClassic Bit
AS
update BookBrief set IsHeadLine= @IsHeadLine where BookId=@BookId
update BookBrief set IsClassic=@IsClassic where BookId=@BookId

```

4. 创建删除存储过程

在删除图书的过程中, 需要调用两个存储过程 getDataReader 和 DelBookBriefByBookId, 其中存储过程 getDataReader 为获取全部图书并以 SqlDataReader 类型的实例返回, 而存储过程 DelBookBriefByBookId 则是根据传入的 BookId 参数删除 BookBrief 表中 BookId 等于传入参数的记录。

存储过程 getDataReader 代码为:

```

CREATE PROCEDURE [dbo].[getDataReader]
/*
(
@parameter1 int = 5,
@parameter2 datatype OUTPUT
)
*/
AS
select BookId, BookName, BookDesc, isHeadLine, isClassic from bookbrief

```

存储过程 DelBookBriefByBookId 代码为:

```

CREATE PROCEDURE dbo.DelBookBriefByBookId
@BookId int
AS
delete from BookBrief where bookid=@BookId

```

5. 相关技能及知识点

在以上存储过程的定义过程中，不难发现有些存储过程执行的是一个插入语句，其返回值是影响到行数，而有些存储过程执行的是一组插入语句。如果系统在执行到该存储过程中的某一条插入语句时，突然断电或系统发生故障停止运转，那么已经执行过的语句应该已经改写了数据库中的数据，而由于故障或停电造成尚未执行的语句则没有执行，则此时该存储过程将自动回滚到尚未执行时的状态，也就是将数据库中的数据回滚至此次操作尚未执行时的状态。因为保存在数据库中的存储过程是事物一级的功能。然而，有些时候因为某种需要可能会在程序代码中直接调用 SQL 语句，也就是直接在程序代码中执行若干条插入、检索语句，那么在这个时候就应该考虑到系统运转过程中所可能遇到的各种应急情况，此时就应该在多条语句执行的地方加入事物的限制，防止出现业务逻辑错误或其他异常现象的发生。

1.11.6 BLL层：图书详情BookBriefManager类的实现

在 BLL 类库中添加新类，命名为 **BookBriefManager**，定义所属命名空间为 **namespace BookShop.BLL**，标记为可序列化[Serializable]，访问权限为 **public**。该类负责收集用于添加新书时的各种图书详情及修改图书热点和经典推荐时的信息，并执行相应的添加图书详情和修改图书热点、经典类型的操作，以及显示当前 **BookBrief** 表中的图书信息，并根据选中的图书项进行删除操作。

```
namespace BookShop.BLL
{
    public class BookBriefManager
    {
        private static readonly BookBriefAccess ba = new BookBriefAccess();

        public void AddBookBrief(BookBriefInfo bbf)
        {
            ba.AddBookBrief(bbf);
        }

        public void UpdateBookBriefByBookId(
            BookBriefInfo bbf)
        {
            ba.UpdateBookBriefByBookId(bbf);
        }
        //获取全部图书
        public SqlDataReader getDataReader()
        {
            return ba.getDataReader();
        }
        //删除指定 BookId 的图书
        public void DelBookBriefByBookId(int id)
        {
```

```

        ba.DelBookBriefByBookId(id);
    }
}
}

```

1.11.7 图书信息管理表示层代码实现

后台图书信息管理表示层包括图书热点和经典管理页面 `BookEdit.aspx` 及其代码文件 `BookEdit.aspx.cs`、添加图书信息管理页面 `AddBook.aspx` 及其代码文件 `AddBook.aspx.cs`、后台图书信息管理页面 `BookManager.aspx` 及其代码文件 `BookManager.aspx.cs`，以及后台图书删除管理页面 `BookDel.aspx` 及其代码文件 `BookDel.aspx.cs`。

1. 图书热点和经典管理页面BookEdit.aspx的HTML代码

```

<% @PageLanguage="C#" MasterPageFile="~/admin/MasterPage.master" AutoEventWireup="true"
CodeFile="BookEdit.aspx.cs" Inherits="admin_BookEdit" Title="无标题页" %>
<asp:ContentID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <table>
        <tr>
            <td>
                <label>输入书号:</label>
                <asp:TextBox ID="txtboxBookId" runat="server" Height="22px"
                    Width="200px" ></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td>
                <label>热点图书:</label>
                <asp:DropDownListID="dropdownIsHeadLine"runat="server"Height="22px"
                    Width="200px" >
                </asp:DropDownList>
            </td>
        </tr>
        <tr>
            <td>
                <label>经典图书:</label>
                <asp:DropDownListID="dropdownIsClassic"runat="server"Height="22px"
                    Width="200px" >
                </asp:DropDownList>
            </td>
        </tr>
        <tr align="center">
            <td>
                <asp:Button ID="btnOk" runat="server" Text="修改" Width="85px"
                    onclick="btnOk_Click" />
                <asp:Button ID="btnCancel" Width="85px"
                    runat="server" Text="返回" onclick="btnCancel_Click" />
            </td>
        </tr>
    </table>

```

</tr>
</table>
</asp:Content>

BookEdit.aspx 页面中各个控件的属性设置及其用途如表 1-47 所示。

表 1-47 BookEdit.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	用 途
asp: TextBox	txtboxBookId	收集图书编号
asp: DropDownList	dropdownIsHeadLine	设定热点推荐类型（是/否）
asp: DropDownList	dropdownIsClassic	设定经典推荐类型（是/否）
asp: Button	btnOk	提交图书信息，进行修改
asp: Button	btnCancel	取消本次操作，返回上一页面

图书修改页效果图如图 1-70 所示。



图 1-70 图书修改页

2. BookEdit.aspx.cs代码文件的实现

```
public partial class admin_BookEdit : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            this.Bind();
        }
    }
    protected void Bind()
    {
        //绑定目录类别下拉列表框
```

```

//绑定是否热点, 是否经典
string[] f = { "是", "否" };
dropdownIsHeadLine.DataSource = f;
dropdownIsHeadLine.DataBind();
dropdownIsClassic.DataSource = f;
dropdownIsClassic.DataBind();
}

protected void btnCancel_Click(object sender, EventArgs e)
{
    Response.Redirect("BookManager.aspx");
}

protected void btnOk_Click(object sender, EventArgs e)
{
    //BookEdit.aspx 根据书号修改热点, 经典类型
    BookBriefManager bbm1 = new BookBriefManager();
    BookBriefInfo bbf1 = new BookBriefInfo();

    bbf1.BookId = txtboxBookId.Text.Trim();
    string isClassic = dropdownIsClassic.SelectedValue;

    if (isClassic.Equals("是"))
    {
        bbf1.IsClassic = true;
    }
    else
    {
        bbf1.IsClassic = false;
    }
    string isHeadLine = dropdownIsHeadLine.SelectedValue;

    if (isHeadLine.Equals("是"))
    {
        bbf1.IsHeadLine = true;
    }
    else
    {
        bbf1.IsHeadLine = false;
    }
    bbm1.UpdateBookBriefByBookId(bbf1);
}
}

```

3. 添加图书信息管理页面AddBook.aspx

AddBook.aspx 页面中各个控件的属性设置及其用途如表 1-48 所示。

表 1-48 AddBook.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
asp:TextBox	txtboxBookId		输入图书编号
	txtboxBookName		输入图书名称
asp:RequiredFieldValidator	RequiredFieldValidator1	ControlToValidate=txtboxBookId	验证图书编号控件是否为空
asp:RequiredFieldValidator	RequiredFieldValidator1	ControlToValidate=txtboxBookName	验证图书名称控件是否为空
asp:TextBox	txtboxBookPrice		填写图书售价
asp:RequiredFieldValidator	RequiredFieldValidator1	ControlToValidate=txtboxBookPrice	验证图书售价控件是否为空
asp:TextBox	txtboxBookQty		填写图书数量
asp:RequiredFieldValidator	RequiredFieldValidator1	ControlToValidate=txtboxBookQty	验证图书数量控件是否为空
asp:DropDownList	dropdownCategoryId	后台填充目录名称项	选择新添加图书的所属类别
asp:DropDownList	dropdownIsHeadLine	后台填充图书热点项	选择新添加图书热点类别
asp:DropDownList	dropdownIsClassic	后台填充图书经典项	选择新添加图书经典类别
asp:Button	btnAddOk		提交数据，更新数据库
asp:Button	Button2		退出添加新书操作，返回上一界面

添加图书管理页面效果图如图 1-71 所示。



图 1-71 添加图书管理页面

4. AddBook.aspx.cs代码文件的实现

```
public partial class admin_AddBook : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            this.Bind();
        }
    }
}
```

//填充页面 Gridview 控件的数据源：目录

```
public void Bind()
{
    //CategoryManager cm = new CategoryManager();
    //gvCategories.DataSource = cm.GetCategoryInfos();
    //gvCategories.DataBind();
    //填充下拉列表框
    CategoryManager cm1 = new CategoryManager();
    dropdownCategoryId.DataSource = cm1.GetCategoryInfos();
    dropdownCategoryId.DataBind();
    string[] f = { "是", "否"};
    dropdownIsHeadLine.DataSource = f;
    dropdownIsHeadLine.DataBind();
    dropdownIsClassic.DataSource = f;
    dropdownIsClassic.DataBind();
}
```

protected void btnAddOk_Click(object sender, EventArgs e)

```
{
    //收集更新 BookBrief 表的数据
    BookBriefInfo bbf = new BookBriefInfo();
    bbf.BookId = txtboxBookId.Text.Trim();
    bbf.CategoryId = dropdownCategoryId.SelectedValue;
    bbf.BookName = txtboxBookName.Text.Trim();

    string isClassic = dropdownIsClassic.SelectedValue;
    if (isClassic.Equals("是"))
    {
        bbf.IsClassic = true;
    }
    else
    {
        bbf.IsClassic = false;
    }

    string isHeadLine = dropdownIsHeadLine.SelectedValue;
    if (isHeadLine.Equals("是"))
    {
        bbf.IsHeadLine = true;
    }
    else
    {
        bbf.IsHeadLine=false;
    }
    //开始更新 BookBrief 表
    BookBriefManager bbm = new BookBriefManager();
    bbm.AddBookBrief(bbf);
}
```

```

//收集 Item 表的数据
ItemInfo info = new ItemInfo();
info.BookId = txtboxBookId.Text.Trim();
info.BookPrice = Decimal.Parse(txtboxBookPrice.Text.Trim());
info.Quantity = Int32.Parse(txtboxBookQty.Text.Trim());
ItemManager im = new ItemManager();
im.AddItem(info);
lblmsg.Text = "添加成功!";
}
protected void btnAddCancel_Click(object sender, EventArgs e)
{
Response.Redirect("BookBriefManager.aspx");
}
}

```

5. 图书信息管理页面BookManager.aspx

BookManager.aspx 页面中各个控件的属性设置及其用途如表 1-49 所示。

表 1-49 BookManager.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
asp:HyperLink	hyAddBook	NavigateUrl="~/admin/AddBook.aspx"	添加图书
	hyBookBrief	NavigateUrl="~/admin/BookEdit.aspx"	修改概要
asp:HyperLink	hyDefault	NavigateUrl="~/admin/Default.aspx"	返回系统
asp:HyperLink	hyBookDel	NavigateUrl="~/admin/BookDel.aspx"	删除图书

图书信息管理页面效果图如图 1-72 所示。



图 1-72 图书信息管理页面

6. 后台删除图书信息管理页面BookDel.aspx

```

<% @PageLanguage="C#" MasterPageFile="~/admin/MasterPage.master" AutoEventWireup="true"
CodeFile="BookDel.aspx.cs" Inherits="admin_BookDel" Title="无标题页" %>

<asp:ContentID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <asp:GridView ID="GridView1" runat="server" AutoGenerateDeleteButton="True"
onrowdeleting="GridView1_RowDeleted">

```



```
</asp:GridView>
</asp:Content>
```

BookDel.aspx 页面中各个控件的属性设置及其用途如表 1-50 所示。

表 1-50 BookDel.aspx 页面中各个控件的属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用 途
asp:GridView	GridView1	AutoGenerateDeleteButton="True"	显示删除按钮
		onrowdeleting="GridView1_RowDeleted"	单击删除按钮时执行 GridView1_RowDeleted 事件

后台删除图书信息管理页面效果图如图 1-73 所示。



图 1-73 后台删除图书信息管理页面

7. BookDel.aspx.cs代码文件的实现

```
public partial class admin_BookDel : System.Web.UI.Page
{
    protected readonly BookBriefManager bbm = new BookBriefManager();
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            this.myDataBind();
        }
    }

    protected void myDataBind()
    {
        GridView1.DataSource = bbm.getDataReader();
        GridView1.DataKeyNames = new string[] { "BookId" };
        GridView1.DataBind();
    }
}
```

```
protected void GridView1_RowDeleted(object sender, GridViewDeleteEventArgs e)
{
    int id = int.Parse(GridView1.DataKeys[e.RowIndex].Value.ToString());
    bbm.DelBookBriefByBookId(id);
    GridView1.DataSource = bbm.getDataReader();
    GridView1.DataKeyNames = new string[] { "BookId" };
    GridView1.DataBind();
}
```

1.11.8 任务小结

在完成图书管理的任务中遵循三层标准架构，主要实现了图书管理的数据访问类、图书管理的业务逻辑类、图书管理的页面及代码等，在对数据访问时主要使用的是存储过程。在删除图书过程中，主要是通过用于呈现图书项目的 GridView 控件中的删除按钮来实现的。

当使用 GridView 进行数据行的删除操作时，需要预先指定 GridView 控件的 DataKeyNames 属性，并且需要在 HTML 视图下设置 GridView 的 AutoGenerateDeleteButton="True" 和 onrowdeleting="GridViewL_RowDeleted" 属性，即自动显示删除链接和当单击删除链接时触发 GridView1_RowDeleted 事件。

在该事件中通过事件参数 GridViewDeleteEventArgs 类型的 e.RowIndex 属性获取行索引值，而该值正是设置 DataKeyNames 属性时所带表的数据表主键字段在该 GridView 控件中的具体行索引。

最后使用 GridView 的 DataKeys[e.RowIndex].Value.ToString()方法，即可获取选中行的主键字段值。

1.11.9 练习题

完善程序：要求通过选择 GridView 控件中某行的删除链接按钮删除该行记录。

```
protected void GridView1_RowDeleted(object sender, GridViewDeleteEventArgs e)
{
    int id = int.Parse(GridView1.DataKeys[_____].Value.ToString());
    bbm.DelBookBriefByBookId(id);
    GridView1.DataSource = bbm.getDataReader();
    GridView1.DataKeyNames = new string[] { "BookId" };
    GridView1.DataBind();
}
```

1.12 生成及发布网站

学习目标

- 掌握如何生成及发布网站
- 掌握如何在本机上试运行
- 建议学时：2 学时

1.12.1 任务名称：生成及发布网站

1.12.2 任务描述

生成网站、发布网站，以及在本机上试运行。

1.12.3 任务分析

VS2008 提供了“生成网站”和“发布网站”的命令，在本机上试运行则需要 IIS 中进行相应的设置。

1.12.4 生成网站及发布网站

① 在“解决方案资源管理器”中右击网站 Web，如图 1-74 所示，首先生成网站；成功后执行发布网站的命令，如图 1-75 所示。



图 1-74 生成网站



图 1-75 发布网站

在打开的“发布网站”对话框中选择网站发布的目标位置，如图 1-76 所示。

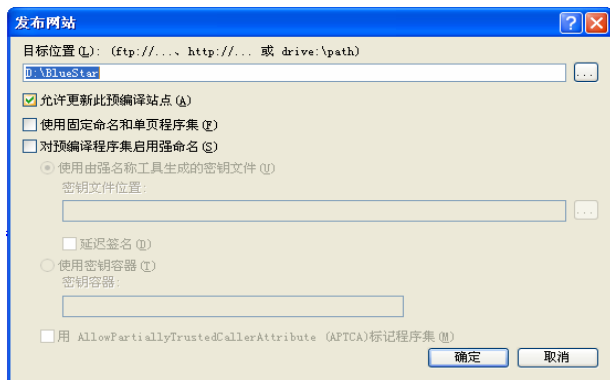


图 1-76 选择位置

② 发布完成后，要在本机上试运行可右击“我的电脑”，选择“管理”，在计算机管理窗口的树状目录中选择“Internet 信息服务”。下面以 XP 系统为例，通过“虚拟目录创建向导”新建虚拟目录，如图 1-77~图 1-80 所示。



图 1-77 新建虚拟目录



图 1-78 输入路径

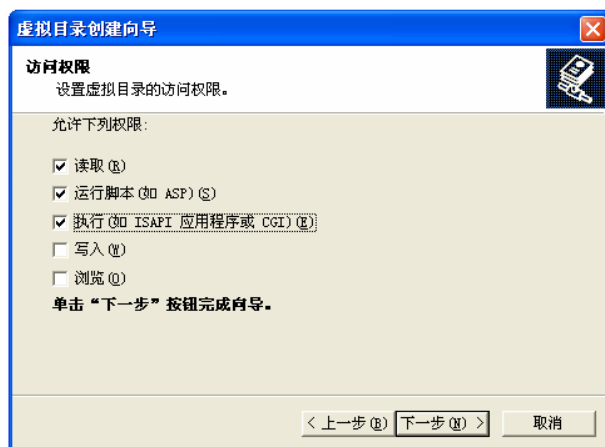


图 1-79 设置访问权限



图 1-80 完成虚拟目录创建

③ 因连接数据库服务器采用的是“Windows 身份验证”，所以还要在 SQL Server 2005 数据库中做如下设置。

首先，在“对象资源管理器”中展开“安全性”目录，再展开下级的“登录名”目录，右击“×××\ASPNET”，选择“属性”；在“登录属性”对话框的“选择页”窗格选择“用户映射”，在右侧对应页上，“映射”勾选自定义数据库“CVITBOOKSHOP”，身份勾选“db_owner”，如图 1-81 所示，单击“确定”按钮。

其次，在“对象资源管理器”中展开数据库“CVITBOOKSHOP”目录，再展开下级的“安全性”目录，继续展开“用户”目录，这时可以看到由刚才操作添加的用户“×××\ASPNET”；右击“×××\ASPNET”，选择“属性”，在弹出的对话框中设置数据库角色成员身份，勾选“db_owner”，如图 1-82 所示。

1.12.5 任务小结

通过本任务的学习，完成了网站的生成与发布，并且在本机上试运行，主要应掌握如何执行相应的命令，以及在数据库服务器中设置登录用户的属性。

1.12.6 练习题

1. 完成网站发布并在本机上试运行。
2. 讨论如何在服务商提供的服务器上部署网站。

第 2 章 企业新闻发布信息管理系统

2.1 用户需求的分析与处理

学习目标

■ 理解需求阶段的目标

■ 给业务上下文和系统功能建模

■ 在完整的用例模型中记录系统需求

■ 建议学时：8 学时

2.1.1 任务名称：用户需求的分析与处理

2.1.2 任务描述

用户需求的分析与处理就是解决“做什么”的问题，就是要全面地理解用户对新闻系统的各项要求，并准确地表达所接受的用户需求。

2.1.3 任务分析

需求分析阶段的工作，可以分为 4 个方面：问题识别，分析与综合，制订规格说明书，评审。

（1）问题识别

问题识别即从系统角度来理解软件，确定对所开发系统的综合要求，并提出这些需求的实现条件，以及需求应该达到的标准。这些需求包括：功能需求（做什么），性能需求（要达到什么指标），环境需求（如机型，操作系统等），可靠性需求（不发生故障的概率），安全保密需求，用户界面需求，资源使用需求（软件运行时所需的内存、CPU等），软件成本消耗与开发进度需求，预先估计以后系统可能达到的目标。

（2）分析与综合

逐步细化所有的软件功能，找出系统各元素间的联系、接口特性和设计上的限制，分析其是否满足需求，剔除不合理部分，增加需要部分。最后，综合成系统的解决方案，给出要开发系统的详细逻辑模型（做什么的模型）。

（3）制订规格说明书

制订规格说明书即编制文档，描述需求的文档称为 软件需求规格说明书。应注意，需求分析阶段的成果是需求规格说明书，向下一阶段提交。

（4）评审

对功能的正确性、完整性和清晰性，以及其他需求给予评价。评审通过才可进行下一阶段的工作，否则需重新进行需求分析。

2.1.4 收集用户需求

1. 项目的背景及意义

当今社会是信息竞争的社会，企业的信息化建设是提高企业管理效率的必经途径，在这样一个信息化建设中，企业的新闻发布系统是企业对外快速传播信息的门户。

这个“门户”让拥有它的企业能够及时发布企业的最新信息，让其他用户第一时间获取信息，以此占有市场先机。谁拥有互联网，谁就拥有了信息；谁拥有了信息，谁就能占据有利竞争地位。这已经成为一条新的市场竞争规则。

本项目分为前、后台管理系统。

前台实现的功能主要包括用户注册、修改已注册用户信息，具体如下：

- 注册用户发布新闻功能；
- 新闻搜索功能；
- 各新闻类别中新闻数量的统计功能；
- 用户对新闻进行评论功能；
- 热点新闻统计及浏览功能；
- 按类别浏览新闻功能。

后台实现的功能主要包括以下部分：

- 管理现有新闻；
- 发布新的新闻；
- 对要发布的新闻进行审核；
- 管理新闻评论；
- 管理新闻栏目；
- 管理系统用户。

2. 企业新闻发布信息管理系统业务流程

企业新闻发布信息管理系统业务流程如图 2-1 所示。

3. 用户需求调查问卷

问卷主要包括以下部分：

- 调查人姓名；
- 调查人所在部门；
- 调查人职务；
- 调查日期；
- 建议软件名称；
- 该软件的使用者、部门、角色、主要任务；
- 与软件运行有关的实体、实体名称、关系；
- 软件工作平台与体系结构的要求；
- 管理系统体系结构；
- 软件开发工具的要求；
- 软件功能上的要求；
- 软件性能上的要求；
- 软件安全方面的要求；
- 软件约束性要求；

➤ 软件使用方便的要求。

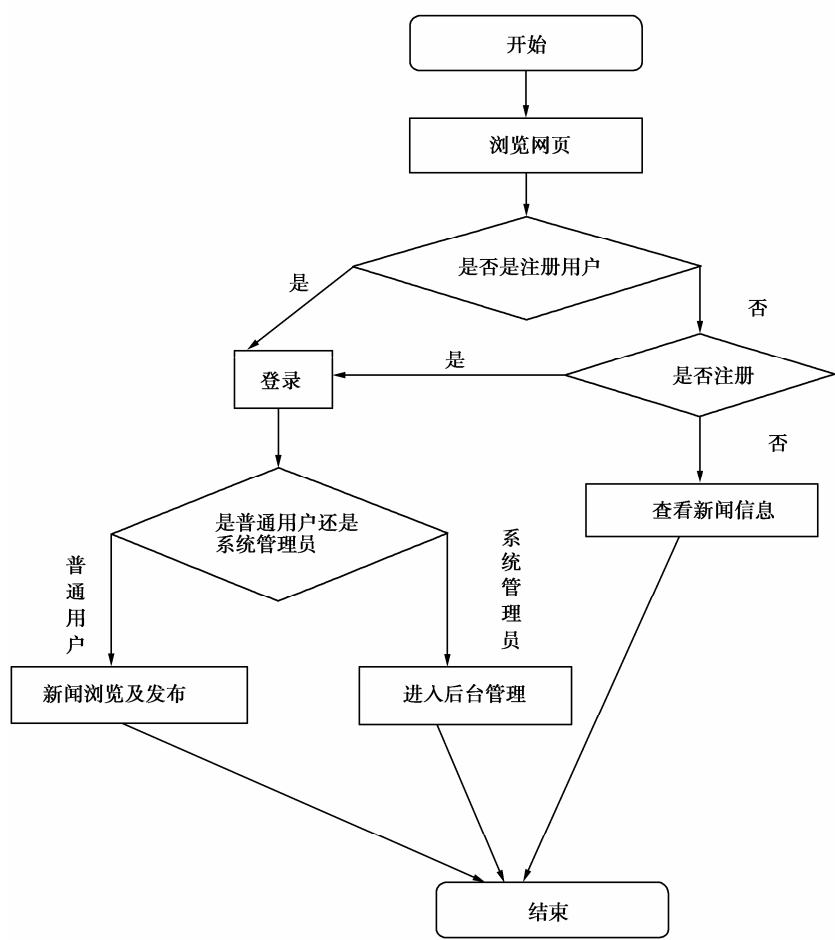


图 2-1 企业新闻发布信息管理系统业务流程

4. 用户需求

用户需求包括以下内容：

- 用户可以匿名浏览新闻信息，但需注册具有用户资格才能发布新闻；
- 用户注册后可以修改个人信息；
- 用户可以按新闻栏目浏览新闻信息；
- 用户可以搜索满足一定条件的新闻；
- 用户在浏览新闻信息时，输入一些必要的个人信息即可对新闻进行评论；
- 用户可以浏览点击率最高的新闻，同时可以知道每条新闻评论的条数及每个新闻栏目新闻的数量；
- 系统注册用户分为普通用户和管理员用户，普通用户可以修改个人信息发布新闻，管理员可以对普通会员资料进行添加与删除；
- 管理员可以创建与维护新闻内容；
- 管理员可以维护新闻评论；
- 系统具有友好性和易操作性；

➤ 系统具有安全性和保密性。

2.1.5 分析用户的需求

1. 功能模型

功能模型如图 2-2 所示。

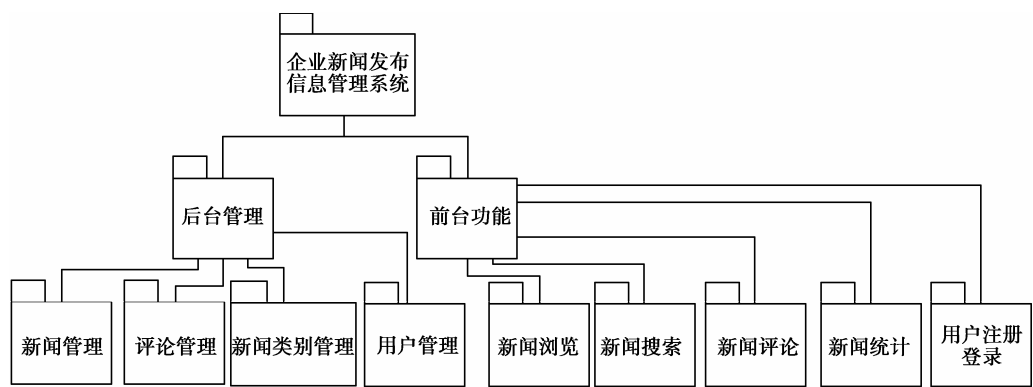


图 2-2 功能模型

2. 撰写需求规格说明书

(1) 产品说明

产品名称：企业新闻发布信息管理系统。

用途：新闻浏览、新闻发布、新闻评论、会员注册等。

产品的开发背景：当今社会是信息竞争的社会，企业的信息化建设是提高企业管理效率的必经途径，在这样一个信息化建设中，企业的新闻发布系统是企业对外快速传播信息的门户，具有重要作用。

该产品的主要优点如下：

- 信息发布及时；
- 运营成本低；
- 增强企业竞争力；
- 增强企业宣传力度，提升企业形象；
- 不受时间、空间限制。

(2) 产品面向的用户群体

产品面向 Internet 上来自全国各地访问该系统的用户。

(3) 产品中的角色

- 管理员：对该系统进行后台维护的工作人员。
- 普通用户：在本系统中注册的用户，可以发布新闻信息。
- 游客：没有在该系统注册，通过 Internet 访问该系统的人员。

(4) 产品的硬件环境要求

- 要求安装有 MS Windows Server 2003/2008 标准版/企业版的服务器。
- 要求安装 IE5.5 以上版本。
- 要求安装 IIS5.0 以上版本。

3. 评审

组织专家组成员评审。对功能的正确性、完整性和清晰性，以及其他需求给予评价。评审通过才可进行下一阶段的工作，否则重新进行需求分析。

2.1.6 任务小结

需求分析和程序设计不尽相同，合理、可行是最重要的。跳出程序设计的圈子，站在系统的角度上看问题，会得到截然不同的结论。

2.1.7 练习题

- 1. 如何进行需求分析？
- 2. 需求分析的意义是什么？

2.2 项目计划安排

学习目标

- 理解软件项目计划的作用与主要组成要素
- 初步掌握如何制定中小型软件项目计划
- 建议学时：4 学时

2.2.1 任务名称：项目计划安排

2.2.2 任务描述

根据 PMBOK2000，项目计划可以包含如下要素：

- 项目范围说明；
- 项目进度计划；
- 项目质量计划；
- 项目资源计划；
- 项目沟通计划；
- 风险对策计划；
- 项目采购计划；
- 变更控制、配置管理计划。

2.2.3 任务分析

《软件项目计划书》的编制可参考《GB 8567—88 计算机软件产品开发文件编制指南》中项目开发计划的要求。各企业在建立 ISO 9001 质量管理体系或 CMM 过程中也会建立相应的《软件开发项目计划书规范》。

编制项目计划的过程应分为以下几个步骤：

- ① 确定项目的应交付成果；
- ② 进行任务分解；
- ③ 确定各个任务开始和结束时间的先后顺序；
- ④ 确定每个任务所需的时间；

- ⑤ 确定项目团队成员可以支配的时间；
- ⑥ 编制项目总体进度计划——任务名称、责任人、开始与结束时间、应提交的可检查的工作成果；
- ⑦ 考虑项目的费用预算、可能的风险。

2.2.4 项目计划

本产品应是一个应用广泛、具有可扩充性和便于维护的企业新闻发布信息管理软件，其包括的内容有：

- 在对企业新闻发布信息管理系统总体业务进行分析的基础上进行提炼，充分考虑系统性与可扩充性；
- 遵循国家及行业规范；
- 便于维护和扩充；
- 系统安全、稳定、可移植性好；
- 采用基于三层框架架构，系统结构灵活。

1. 应交付成果

本产品应交付的成果如下：

- 系统源程序；
- 系统需求规格说明书；
- 系统使用说明书。

2. 任务分解及具体安排

任务分解及具体安排如表 2-1 所示。

表 2-1 任务分解及具体安排

系 统 名 称	工 作	所 需 人 员	所 需 天 数	标志性事件（交付物）
企业新闻发布信息 管理系统	需求分析	2	3	需求规格说明书
	系统设计	2	3	概要设计说明书
	子系统设计	3	6	详细设计说明书
	系统编码与实现	3	14	可测试代码
	系统测试	4	2	测试报告
	系统实施与维护、使用培训	2	3	用户手册 验收报告

3. 项目计划安排的审核

项目计划书评审、批准是为了使相关人员达成共识，减少不必要的错误，使项目计划更合理、有效。

在企业中，项目经理完成《软件项目计划书》后，首先组织项目团队内部的项目团队负责人、测试负责人、系统分析负责人、设计负责人、质量监督员等项目计划书进行评审，并进行阶段成果项目团队内评阅记录。应要求所有相关人员在收到软件项目计划书后的一个约定时间内反馈对计划书的意见。项目经理确保与所有人员就项目计划书中所列内容达成一致。这种一致性是要求所有项目团队成员对项目计划的内容进行承诺，无法承诺或者说无法达成一致的，要么修改项目计划以适应某些项目团队成员，要么由某些项目团队成员采取妥协措施，适应项目计划的要求。

项目经理将已经达成一致的软件项目计划书提交项目高层分管领导或其授权人员进行审批，审批完成时间不能超过预先约定的时间。对于意义重大的项目，由过程控制部门如质量管理部和项目分管领导同时对《软件项目计划书》进行审批。

批准后的《软件项目计划书》作为项目活动开展依据和本企业进行项目控制与检查的依据，并在必要时根据项目进展情况实施计划变更。

项目质量监督员根据《软件项目计划书》和《软件开发项目质量计划书规范》编制软件开发项目质量计划。大型的项目应当编制单独的《软件开发项目质量计划书》；规模较小的可以在《软件项目计划书》的某个章节说明“软件开发项目质量计划”，也可单独编制类似“软件开发项目质量控制表”的文档。

配置管理员根据计划书编制《项目配置管理计划》。以项目工作计划书中的阶段成果为依据，根据配置管理计划规范编制配置管理计划，项目经理审批配置管理计划，并对配置管理计划的有效性负责。

2.2.5 任务小结

为了保证项目团队按时、保质地完成项目目标，便于项目团队成员更好地了解项目情况，使项目工作开展的各个过程合理、有序，以文件化的形式，把对于在项目生命周期内的工作任务范围、各项工作的任务分解、项目团队组织结构、各团队成员的工作责任、团队内外的沟通协作方式、开发进度、项目内外环境条件等内容做出的安排以书面的方式，作为项目团队成员及项目干系人之间的共识与约定、项目生命周期内所有项目活动的行动基础，以及项目团队开展和检查项目工作的依据。

2.2.6 练习题

项目计划的意义是什么？

2.3 系统设计

学习目标

■ 理解系统架构的主要作用

■ 初步掌握系统架构的设计方法

■ 建议学时：4 学时

软件架构是指一个系统的基础组织，具体体现在系统的组成构件，构件之间、构件和环境之间的关系，以及指导其设计和演化的原则上。

2.3.1 任务名称：系统架构设计

2.3.2 任务描述

完成企业新闻发布信息管理系统架构。

2.3.3 任务分析

架构由许多不同的架构视图来表示，这些视图本质上是以图形方式来摘要说明“在架构方面具有重要意义”的模型元素。在 **Rational Unified Process** 中，将从一个典型的视图集开始，该视图集称为“4+1 视图模型”[KRU95]，它包括以下内容。

- ① 用例视图：包括用例和场景，这些用例和场景包括在架构方面具有重要意义的行为、类或技术风险。它是用例模型的子集。
- ② 逻辑视图：包括最重要的设计类、从这些设计类到包和子系统的组织形式，以及从这些包和子系统到层的组织形式；还包括一些用例实现。它是设计模型的子集。
- ③ 实施视图：包括实施模型及其从模块到包和层的组织形式的概览；同时，还描述了将逻辑视图中的包和类向实施视图中的包和模块分配的情况。它是实施模型的子集。
- ④ 进程视图：包括所涉及任务（进程和线程）的描述、它们的交互和配置，以及将设计对象和类向任务的分配情况。只有在系统具有很高程度的并行时，才需要该视图。在 **Rational Unified Process** 中，它是设计模型的子集。

⑤ 配置视图：包括对最典型平台配置的各种物理节点的描述及将任务（来自进程视图）向物理节点分配的情况。只有在分布式系统中才需要该视图，它是部署模型的一个子集。

架构视图记录在软件架构文档中。可以构建其他视图来表达需要特别关注的不同方面：用户界面视图、安全视图、数据视图等。对于简单的系统，可以省略 4+1 视图模型中的一些视图。

企业新闻发布信息管理系统架构设计要达到如下的目标。

- 可靠性（Reliable）。软件系统对于用户的商业经营和管理来说极为重要，因此软件系统必须非常可靠。
- 安全性（Secure）。软件系统所承担交易的商业价值极高，系统的安全性非常重要。
- 可伸缩性（Scalable）。软件必须能够在用户的使用率、用户的数目增加很快的情况下，保持合理的性能。只有这样，才能适应用户市场扩展的可能性。
- 可定制化（Customizable）。同样的一套软件，可以根据客户群的不同和市场需求的变化进行调整。
- 可扩展性（Extensible）。在新技术出现的时候，一个软件系统应当允许导入新技术，从而对现有系统进行功能和性能的扩展。
- 可维护性（Maintainable）。软件系统的维护包括两个方面，一是排除现有的错误，二是将新的软件需求反映到现有系统中去。一个易于维护的系统可以有效地降低技术支持的成本。
- 客户体验（Customer Experience）。软件系统必须易于使用。
- 市场时机（Time to Market）。软件用户要面临同业竞争，软件提供商也要面临同业竞争，以最快的速度争夺市场先机非常重要。

2.3.4 架构重点及模式

虽然以上视图可以表示系统的整体设计，但架构只和以下几个具体方面相关：

- 模型的结构，即组织模式，如分层；
- 基本元素，即关键用例、主类、常用机制等，它们与模型中的各元素相对应；
- 几个关键场景，它们表示了整个系统的主要控制流程。

架构视图在本质上是整体设计的抽象或简化，它们通过舍弃具体细节来突出重要的特征。在考虑以下方面时，这些特征非常重要：

- 系统演进，即进入下一个开发周期；
- 在产品线环境下复用架构或架构的一部分；
- 评估补充质量，如性能、可用性、可移植性和安全性；
- 向团队或分包商分配开发工作；
- 决定是否包括市售构件；
- 插入范围更广的系统。

架构模式是解决复发架构问题的现成形式。架构框架或架构基础设施（中间件）是可以在其上构建某种架构的构件集。许多主要的架构困难应在框架或基础设施中予以解决，而且通常针对特定的领域——命令和控制、MIS、控制系统等。

架构视图的图形描述称为架构设计图。对于以上描述的各种视图，设计图由以下统一的建模语言图组成 [UML99]。

逻辑视图：类图、状态机和对象图。

进程视图：类图与对象图（包括任务、进程与线程）。

实施视图：构件图。

部署视图：配置图。

用例视图：用例图描述用例、主角和普通设计类；顺序图描述设计对象及其协作关系。

架构设计流程：在 Rational Unified Process 中，架构主要是分析设计工作流程的结果。当项目再次进行此工作流程时，架构将在一次又一次迭代中不断演化、改进、精炼。由于每次迭代都包括集成和测试，所以在交付产品时，架构已经相当强壮。架构是精化阶段各次迭代的重点，架构的基线通常会在此阶段结束时确定。

2.3.5 选择技术

开发环境：Microsoft Visual Studio 2008 集成开发环境。

编程语言：ASP.NET+C#。

数据库：Microsoft SQL Server 2005。

2.3.6 安全策略

企业新闻发布信息管理系统的实施，其关键是要保证新闻的合法性及系统的安全性。所有新闻的发布均要通过管理员的审核以后才能发布，用户的密码则采用密码技术，通过 MD5 加密算法保证安全性。同时，保证数据库的及时备份。

2.3.7 任务小结

软件体系结构是构建计算机软件实践的基础。软件架构是有关如下问题的设计层次：在计算的算法和数据结构之外，设计并确定系统整体结构。结构问题包括总体组织结构和全局控制结构；通信、同步和数据访问的协议；设计元素的功能分配；物理分布；设计元素的组成；定标与性能；备选设计的选择。架构不仅是结构，还包括符合系统完整性、经济约束条件、审美需求和样式等内容。它不仅注重对内部的考虑，而且还在系统的用户环境和开发环境中对系统进行整体考虑，即同时注重对外部的考虑。

2.3.8 练习题

如何进行系统设计？系统设计有什么意义？

2.4 子系统设计

学习目标

■ 理解如何设计业务层

■ 建议学时：8 学时

2.4.1 任务名称：子系统设计

2.4.2 任务描述

完成企业新闻发布信息管理系统子系统设计。

2.4.3 任务分析

通过子系统设计形成一个实用、合理的解决方案。该任务包括：

- 类的设置；
- 数据库的设置。

2.4.4 类的列表

类的列表如表 2-2 所示。

表 2-2 类的列表

包		类 名	说 明
Web	adminManager	Default	前台首页页面类
		BigTypeNews	前台新闻栏目页面类
		ListView.aspx	前台新闻内容浏览及评论页面类
		MoreComments	前台新闻全部评论浏览页面类
		AllNews	前台全部新闻页面类
		Search	前台新闻搜索页面类
		UserReg	前台用户注册页面类
		UserAddNews	前台用户发布新闻页面类
		UserCenter	前台个人管理信息页面类
		Admin_Login	后台登录页面类
		Admin_Index	后台首页页面类
		Admin_NewsList	后台管理现有新闻页面类
		Admin_EditNews	后台修改新闻页面类
		Admin_DeleteNews	后台删除新闻页面类

包		类 名	说 明
Web	adminManager	Admin_AddNews	后台发布新闻页面类
		Admin_CheckNews	后台审核新闻页面类
		CheckNews	管理审核功能页面类
		Admin_Comments	后台管理新闻评论页面类
		Admin_BigClass	后台 管理新闻类别页面类
		Admin_EditBig	后台 修改新闻类别页面类
		Admin_DeleteBig	后台 删除新闻类别页面类
		Admin_AllUsers	后台 管理系统用户页面类
		Admin_EditUser	后台 修改用户信息页面类
		Admin_DeleteUser	后台 删除用户信息页面类
BLL	NewsLogic	新闻信息管理逻辑类	
	BigClassLogic	新闻类别管理逻辑类	
	CommentsLogic	新闻评论管理逻辑类	
	UserLogic	用户管理逻辑类	
DAL	DBbase	数据库操作类	
	NewsAccess	新闻数据访问类	
	BigClassAccess	新闻类别数据访问类	
	CommentsAccess	新闻评论数据访问类	
	UserAccess	用户数据访问类	
	FormatString	截取字符串类	
Model	NewsInfo	新闻信息类	
	BigClassInfo	新闻类别信息类	
	CommentsInfo	新闻评论信息类	
	UserInfo	用户信息类	

2.4.5 数据库设计

企业新闻发布信息管理系统采用 SQL 2005 数据库系统。在该系统中新建一个数据库，将其命名为 news2008，然后在该数据库中创建 4 个数据表，分别为新闻信息表（tb_News）、新闻评论表（tb_Comments）、新闻类别表（tb_BigClass）和用户信息表（tb_User）。

1. 各表的含义

本系统数据库中各表的含义如表 2-3 所示。

表 2-3 各表的含义

表 名	说 明
tb_News	新闻内容的详细信息——新闻名、发布时间等

表 名	说 明
tb_Comments	新闻评论的详细信息——评论内容等
tb_BigClass	新闻类别的详细信息
tb_User	用户的详细信息

2. 各表的详细设置

(1) tb_News（新闻信息表）

新闻信息表主要用来保存新闻的基本信息，tb_News 数据表的结构如表 2-4 所示。

表 2-4 新闻信息表（tb_News）的结构

字 段 名 称	类 型	大 小	是 否 为 空	描 述
N_id	int	4	否	新闻 ID（自增主键）
Title	varchar	50	是	新闻标题
Info	text	16	是	新闻内容
BigClassID	varchar	50	是	新闻分类 ID
UserName	varchar	50	是	新闻编辑人姓名
InfoTime	datetime	8	是	上传新闻时间
Hit	int	4	是	新闻点击率（默认值：0）
Flag	varchar	50	是	是否通过审核（默认值：未审核）
Cindex	int	4	是	该新闻的索引

(2) tb_Comments（新闻评论表）

新闻评论表主要用来保存新闻评论的相关信息，tb_Comments 数据表的结构如表 2-5 所示。

表 2-5 新闻评论表（tb_Comments）的结构

字 段 名 称	类 型	大 小	是 否 为 空	描 述
C_id	int	4	否	评论 ID（自增主键）
C_user	varchar	50	是	评论者姓名
C_qq	varchar	50	是	评论者 QQ
C_email	varchar	50	是	评论者邮箱
C_word	varchar	200	是	评论内容
C_time	datetime	8	是	评论时间
NewsID	int	4	是	评论的新闻在 tb_News 中的 N_id 值
Cindex	int	4	是	对同一新闻评论的索引值（如同一新闻有两条评论，则其值按先后顺序为 1，2）

(3) `tb_BigClass`（新闻类别表）
新闻类别表主要用来保存新闻类别的相关信息，`tb_BigClass` 数据表的结构如表 2-6 所示。

表 2-6 新闻类别表（`tb_BigClass`）的结构

字段名称	类型	大小	是否为空	描述
B_id	int	4	否	新闻分类 ID（自增主键）
Name	varchar	50	是	新闻类别名称
Flag	char	10	是	是否显示分类标记（默认值：显示）
Cindex	int	4	是	新闻类别索引
NewsCount	int	4	是	每类新闻对应的新闻总数（默认值：0）

(4) `tb_User`（用户信息表）
用户信息表主要用来保存用户的相关信息，`tb_User` 数据表的结构如表 2-7 所示。

表 2-7 新闻用户信息表（`tb_User`）的结构

字段名称	类型	大小	是否为空	描述
U_id	int	4	否	用户 ID（自增主键）
UserName	varchar	50	否	用户姓名
Password	varchar	50	否	用户密码
E-mail	varchar	50	是	用户邮箱
Level	varchar	50	否	用户级别（分为普通用户和管理员）

系统安装配置后，采取定期完全数据备份与联机数据备份相结合的备份方案，定期备份数据。

2.4.6 任务小结

进行子系统设计要遵循系统设计规范；在设计数据库时要针对系统需求合理设置字段。

2.4.7 练习题

类的设计原则是什么？

2.5 新闻信息显示与检索实现

学习目标

- 掌握三层架构的层次结构及调用
- 掌握数据库的连接及操作
- 掌握 SQL 命令的使用
- 掌握类及方法的实现
- 建议学时：6 学时

2.5.1 任务名称：新闻信息显示与检索实现

2.5.2 任务描述

新闻信息显示与检索是本系统的重要功能之一。该任务包括新闻内容的显示与检索及新闻栏目的显示与检索两部分。本系统采用标准的三层架构。在该任务中，Model 类库中的类 NewsInfo、BigClassInfo 主要完成对数据库中新闻内容表 tb_News、新闻栏目表 tb_BigClass 里的字段的定义；DAL 类库中的类 NewsAccess、BigClassAccess 主要完成对新闻内容及标题操作的各种功能的具体实现；BLL 类库中的类 NewsLogic、BigClassLogic 则完成对 DAL 类库中类的逻辑调用。因此，要完成新闻信息显示与检索这一任务，就要对任务功能进行分析，完成各个类库中类的实现。

2.5.3 任务分析

在本系统中采用标准的三层架构，这三层架构是完成系统前台、后台功能的基础。新闻内容的显示与检索，需要完成实体层 Model 类库中的 NewsInfo 类、数据访问层 DAL 类库中的 NewsAccess 类、逻辑层 BLL 类库中的 NewsLogic 类。其中，NewsInfo 类中定义的属性对应新闻内容表 tb_News 中的字段；NewsAccess 类用于实现新闻内容显示与索引的基本方法；NewsLogic 类则用于完成对 NewsAccess 类的逻辑调用，从而实现新闻内容显示与检索的功能。新闻栏目的显示与检索，要完成实体层 Model 类库中的 BigClassInfo 类、数据访问层 DAL 类库中的 BigClassAccess 类、逻辑层 BLL 类库中的 BigClassLogic 类。其中，BigClassInfo 类中定义的属性对应新闻栏目表 tb_BigClass 中的字段；BigClassAccess 类用于实现新闻栏目显示与索引的基本方法；BigClassLogic 类则用于完成对 BigClassAccess 类的逻辑调用，从而实现新闻栏目显示与检索的功能。

在本系统功能的实现上，需要对数据库进行操作，因此 DAL 层的 DBbase 类是完成这些功能的必要前提。在系统中需要对字符串的长度进行控制，DAL 层的 FormatString 类则用于完成此功能。

本任务完成类的顺序如下：

- ① Model 类库中的 NewsInfo 类、BigClassInfo 类；
- ② DAL 类库中的 DBbase 类、NewsAccess 类、BigClassAccess 类；
- ③ BLL 类库中的 NewsLogic、BigClassLogic 类。

下面按照分析的思路、方法来实现新闻信息显示与检索。

建立一个项目，要先建立解决方案。

① 启动 Visual Studio 2008。选择“文件”→“新建”→“项目”，在弹出的“新建项目”窗口中，单击“项目类型”下“其他项目类型”选项中的“Visua Studio 解决方案”，然后单击右侧“模板”中的“空白解决方案”，接下来设置项目名称及位置。

② 在空白解决方案上右击，选择“添加”→“新建项目”。在弹出的窗口中选择类库并将其命名为“Model”，单击“确定”按钮。

③ 按上面步骤依次添加 DAL 类库和 BLL 类库。

在完成 Model 类库后，新建 DAL 类库时需要在 DAL 类库中添加对 Model 类库的引用。即在 DAL 类库名上单击鼠标右键，选择“添加引用”，在弹出的窗口中单击“项目”选项卡，然后选择项目名称“Model”，单击“确定”按钮。

同样，在建立 BLL 类库时需要添加对 Model 类库和 DAL 类库的引用。
而在新建立网站时需要添加对 Model 类库、DAL 类库、BLL 类库的引用。

2.5.4 Model层：实体类实现

1. NewsInfo类实现

程序开发步骤如下：

① 在 Model 上右击，选择“添加”→“新建项目”，在打开的窗口中选择模板中的“类”，在“名称”栏输入“NewsInfo.cs”，如图 2-3 所示，单击“添加”按钮。

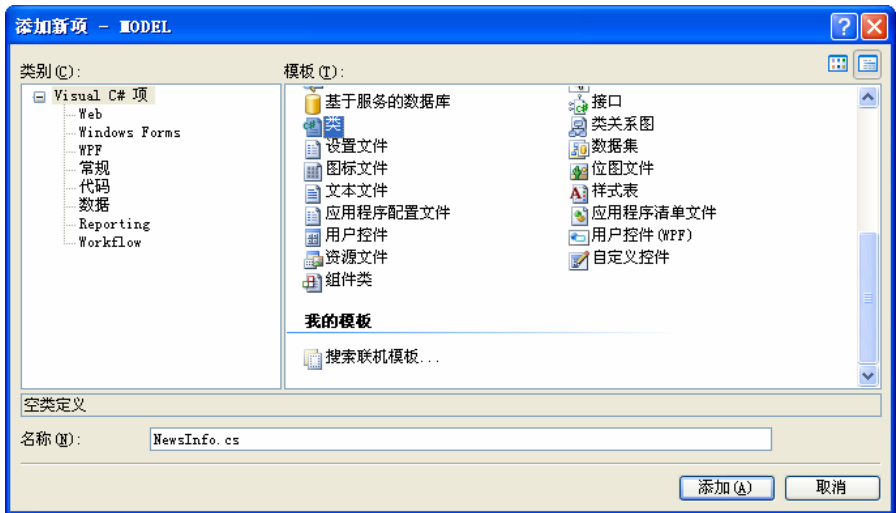


图 2-3 添加 NewsInfo 类

添加后双击“NewsInfo.cs”文件进行编码。定义类所属命名空间为 namespace Model，类的访问修饰符设为 public。NewsInfo 类中主要进行属性的设置，各个属性对应数据库 tb_News 表中的相应字段。

② 完成主要程序代码。NewsInfo 类包含 9 个内部变量，其代码如下。

新闻列表实体类：NewsInfo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Model
{
    /// <summary>
    /// 实体层——新闻列表
    /// 定义成员变量及属性
    /// </summary>
    public class NewsInfo
    {
        #region 定义 news 类的成员变量
```

```

/// <summary>
/// 定义 news 类的成员变量
/// </summary>
private int _n_id;
private string _title;           //新闻标题
private string _info;           //新闻内容
private int _BigClassID;        //栏目 ID
private string _username;       //发布者名称
private string _infotime;       //发布时间
private int _hit;               //点击率
private int _flag;              //是否通过审核,1 通过,0 未通过
private int _cindex;            //最大索引,得到最大索引就得到了新闻总数
#endregion

#region 定义 news 类成员变量的属性
/// <summary>
/// 定义 news 类成员变量的属性
/// </summary>
public int N_id
{
    get { return _n_id; }
    set { _n_id = value; }
}
public string Title
{
    get { return _title; }
    set { _title = value; }
}
public string Info
{
    get { return _info; }
    set { _info = value; }
}
public int BigClassID
{
    get { return _BigClassID; }
    set { _BigClassID = value; }
}
public string UserName
{
    get { return _username; }
    set { _username = value; }
}
public string Infotime
{
    get { return _infotime; }
    set { _infotime = value; }
}

```

```

    }
    public int Hit
    {
        get { return _hit; }
        set { _hit = value; }
    }

    public int Flag
    {
        get { return _flag; }
        set { _flag = value; }
    }
    public int Cindex
    {
        get { return _cindex; }
        set { _cindex = value; }
    }
    #endregion
}
}

```

2. BigClassInfo类实现

程序开发步骤如下：

① 在 **Model** 类库中添加新项，选择类，取名为 **BigClassInfo**。添加后双击“**BigClassInfo.cs**”文件进行编码。定义所属命名空间为 **namespace Model**，类的访问修饰符设为 **public**。**BigClassInfo** 类中主要进行属性的设置，各个属性对应数据库 **tb_News** 表中的相应字段。

② 完成主要程序代码。**BigClassInfo** 类包含 5 个内部变量，其代码如下。

新闻栏目列表实体类：**BigClassInfo**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Model
{
    /// <summary>
    /// 实体层——新闻栏目表
    /// 定义成员变量及属性
    /// </summary>
    public class BigClassInfo
    {
        #region 定义类的成员变量
        /// <summary>
        /// 定义类的成员变量
        /// </summary>
        private int _b_id;//新闻类别 ID
    }
}

```



```

private string _name;//新闻栏目名称
private string _flag; //是否显示分类标记
private int _cindex; //最大索引代表新闻类别的总数
private int _newscount;//每类新闻所包含的新闻总数
#endregion

#region 定义成员变量的属性
/// <summary>
/// 定义类的成员变量的属性
/// </summary>
public int B_id
{
    get { return _b_id; }
    set { _b_id = value; }
}
public string Name
{
    get { return _name; }
    set { _name = value; }
}
public string Flag
{
    get { return _flag; }
    set { _flag = value; }
}
public int Cindex
{
    get { return _cindex; }
    set { _cindex = value; }
}
public int Newscount
{
    get { return _newscount; }
    set { _newscount = value; }
}
#endregion
}
}

```

2.5.5 DAL层：数据访问类实现

1. DBbase.cs类实现

程序开发步骤如下：

- ① 在 DAL 类库中添加新项，选择类，取名为 DBbase。添加后双击“DBbase.cs”文件进行编码。定义所属命名空间为 namespace DAL，类的访问修饰符设为 public。
- ② 完成主要程序代码。DBbase.cs 类主要实现数据库连接及对 SQL 命令的执行，其代码

如下。

数据库的数据访问类：DBbase.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace DAL
{
    /// <summary>
    /// DBbase 的摘要说明
    /// </summary>
    public class DBbase
    {
        public DBbase()
        {
        }

        public static string strCon = System.Configuration.ConfigurationSettings.AppSettings
["conStr"].ToString();//定义连接字符串 strCon
```

在定义完 strCon 数据成员后，需要在企业新闻发布信息管理系统网站的 web.config 文件中添加<appSettings>配置节，定义 conStr 的值。代码如下：

```
<appSettings>
    <!-- 连接 SQL Server 2005 数据库用的连接字符串，其中 data source 的值是服务器名称，uid
的值是登录 SQL Server 身份验证的登录名，pwd 的值是密码，database 的值是要操作的数据库的名称 -->
    <add key="conStr" value="data source=.;uid=sa;pwd=123456;database=news2008;pooling=
true"></add>
</appSettings>

SqlConnection con = new SqlConnection(strCon); //实例化连接对象 con
//检测连接的方法 CheckConnection(),若连接是关闭的则打开 SqlConnection 连接
public void CheckConnection()
{
    if (this.con.State == ConnectionState.Closed)
    {
        this.con.Open();
    }
}

public DataSet ReturnDataSet(string strSQL) //执行语句返回 DataSet 数据集
{
    CheckConnection();
    try
    {
        SqlDataAdapter sda = new SqlDataAdapter(strSQL, con);
        DataSet ds = new DataSet();
```

```

        sda.Fill(ds);
        return ds;
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
    finally
    {
        con.Close();
    }
}

public DataRow GetDataRow(string strSQL) //执行语句返回 DataRow
{
    CheckConnection();
    try
    {
        SqlDataAdapter sda = new SqlDataAdapter(strSQL, con);
        DataSet ds = new DataSet();
        sda.Fill(ds);
        return ds.Tables[0].Rows[0];
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
    finally
    {
        con.Close();
    }
}

```

//执行存储过程或 SQL 语句的方法 ExecuteNonQuery(),执行成功返回 true,否则返回 false

```

public bool ExecuteNonQuery(bool IsPro, string strSQL)
{

```

```

    CheckConnection();
    try
    {
        SqlCommand com = new SqlCommand(strSQL, con);
        if (IsPro)
        {
            com.CommandType = CommandType.StoredProcedure;
        }
        else
        {
            com.CommandType = CommandType.Text;
        }
        com.CommandText = strSQL;
    }

```

```

        com.ExecuteNonQuery();
        con.Close();
        return true;
    }
    catch
    {
        return false;
    }
}

public void ExecuteNonQuery(string strSQL) //执行 SQL 语句的方法 ExecuteNonQuery()
{
    CheckConnection();
    try
    {
        SqlCommand com = new SqlCommand(strSQL, con);
        com.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
    finally
    {
        con.Close();
    }
}

public DataTable ReturnTable(string strSQL) //执行语句返回 DataTable 的方法
{
    CheckConnection();
    try
    {
        SqlDataAdapter sda = new SqlDataAdapter(strSQL, con);
        DataSet ds = new DataSet();
        sda.Fill(ds);
        return ds.Tables[0];
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
    finally
    {
        con.Close();
    }
}

public SqlDataReader ReturnDataReader(string strSQL) //执行语句返回 SqlDataReader 对象
{

```

```

        CheckConnection();
    try
    {
        SqlCommand com = new SqlCommand(strSQL, con);
        SqlDataReader myReader = com.ExecuteReader();
        return myReader;
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
}

public int ReturnRowCount(string strSQL) //执行语句,返回该语句查询出的数据行的总数
{
    CheckConnection();
    try
    {
        SqlDataAdapter da = new SqlDataAdapter(strSQL, con);
        DataSet ds = new DataSet();
        da.Fill(ds);
        return ds.Tables[0].Rows.Count;
    }
    catch
    {
        return 0;
    }
}
#endregion
}
}

```

2. NewsAccess类实现

程序开发步骤如下：

① 在 DAL 类库中添加新项，选择类，取名为 NewsAccess。添加后双击“NewsAccess.cs”文件进行编码。定义所属命名空间为 namespace DAL，类的访问修饰符设为 public。

② 完成主要程序代码。NewsAccess.cs 类主要通过构造 SQL 语句及调用 DBbase 类中的方法，实现关于新闻信息的显示与检索。其代码如下。

新闻列表的数据访问类：NewsAccess.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

```

```

namespace DAL
{
    #region 数据访问类
    /// <summary>
    /// 新闻列表的数据访问类
    /// 构造 SQL 语句及调用方法执行
    /// </summary>
    #endregion

    public class NewsAccess
    {
        DBbase db = new DBbase();//实例化DBbase类的对象,用于调用其内部方法来执行不同的
                                操作
        public DataSet GetNewsID()//获取现有新闻 ID
        {
            string strSQL = "select N_id from tb_News where Flag='已审核'";
            return db.ReturnDataSet(strSQL);
        }
        public DataSet GetBigClassIDByNewsID(int id) //获取现有新闻 N_id 获得 BigClassID
        {
            string strSQL = "select BigClassID from tb_News where N_id="+id;
            return db.ReturnDataSet(strSQL);
        }
        public DataSet GetData_news()//获取全部新闻内容
        {
            string strSQL = "select tb_News.*,tb_BigClass.Name from tb_News,tb_BigClass where
            tb_News.Flag='已审核' and tb_News.BigClassID=tb_BigClass.B_id";
            return db.ReturnDataSet(strSQL);
        }
        public DataSet GetData_news(int BigClassID) //获取全部新闻内容
        {
            string strSQL = "select tb_News.*,tb_BigClass.Name from tb_News,tb_BigClass where
            tb_News.Flag='已审核' and tb_News.BigClassID=" + BigClassID + "and tb_News.
            BigClassID= tb_BigClass.B_id";
            return db.ReturnDataSet(strSQL);
        }
        public DataSet GetDataByBigClass(int BigClassID) //分栏目获取新闻列表
        {
            string strSQL = "select * from tb_News where BigClassID =" + BigClassID + " and Flag=
            '已审核' order by Cindex";
            return db.ReturnDataSet(strSQL);
        }
        public DataSet GetDataByBigClassTopN(int BigClassID,int n) //获取每个分栏前 n 条新闻
        {
            string strSQL = "select top "+n+" * from tb_News where BigClassID=" + BigClassID +
            " and Flag='已审核' order by Cindex DESC ";
        }
    }
}

```

```

        return db.ReturnDataSet(strSQL);
    }
    public DataSet GetDataNewN(int n) //获取最新 n 条新闻
    {
        string strSQL = "select top "+n+" N_id,Title,InfoTime,Hit from tb_News where Flag=
'已审核' order by Cindex DESC";
        return db.ReturnDataSet(strSQL);
    }
    public DataSet GetDataTopNHits()//获取点击率最高的 n 条新闻标题
    {
        string strSQL = "select top "+n+" N_id,Title,Hit from tb_News where Flag='已审核'
order by Hit DESC";
        return db.ReturnDataSet(strSQL);
    }
    public bool AddNews(Model.NewsInfo M_news) //添加新闻,添加成功返回 true,否则返回
false
    {
        string times = System.DateTime.Now.ToString();
        int cindex =GetMaxCindex() + 1;
        string strSQL = "insert tb_news (Title,Info,BigClassID,UserName,InfoTime,Cindex)
values('"+ M_news.Title + "','"+ M_news.Info + "','"+ M_news.BigClassID + "','"+
M_news.UserName + "','"+ times + "','"+ cindex + "')";
        return db.ExecuteNonQuery(false, strSQL);
    }
    //修改指定的新闻,修改成功返回 true,否则返回 false
    public bool UpdateNews(Model.NewsInfo M_news)
    {
        string strSQL = "update tb_News set Title='"+ M_news.Title + "','BigClassID=" +
M_news.BigClassID + "','Info='"+ M_news.Info + "','UserName='"+ M_news.UserName + "'
where N_id=" + M_news.N_id;
        return db.ExecuteNonQuery(false, strSQL);
    }
    public bool DeleteNewByID(int id) //根据新闻 id 删除新闻,删除成功返回 true,否则返回
false
    {
        string strSQL = "delete from tb_News where N_id=" + id;
        return db.ExecuteNonQuery(false, strSQL);
    }
    public void UpdateHits(int id) //更新新闻的点击率
    {
        string strSQL = "update tb_News set Hit=Hit+1 where N_id=" + id;
        db.ExecuteNonQuery(strSQL);
    }
    public DataSet DataBindNews(int id) //根据指定 id 绑定指定的新闻内容
    {
        string strSQL = "select * from tb_News where N_id=" + id;
        return db.ReturnDataSet(strSQL);
    }

```

```

    }
    public DataSet QueryByNewsTitle(string title) //按新闻标题模糊查询新闻列表
    {
        string strSQL = "select N_id,Title,InfoTime,Hit from tb_News where Flag='已审核' and
        Title like '%" + title + "%' order by cindex";
        return db.ReturnDataSet(strSQL);
    }
    public DataSet AdminQueryByNewsTitle(string title) //按新闻标题模糊查询新闻列表
    {
        string strSQL = "select tb_News.*,tb_BigClass.Name from tb_News,tb_BigClass where
        tb_News.Flag='已审核' and  tb_News.BigClassID=tb_BigClass.B_id and Title like '%" +
        title + "%' order by tb_News.Cindex";
        return db.ReturnDataSet(strSQL);
    }
    public DataSet QueryByNewsInfo(string info) //按新闻内容模糊查询新闻列表
    {
        string strSQL = "select N_id,Title,InfoTime,Hit from tb_News where Flag='已审核' and
        Info like '%" + info + "%' order by cindex";
        return db.ReturnDataSet(strSQL);
    }
    public DataSet AdminQueryByNewsInfo(string info) //按新闻内容模糊查询新闻列表
    {
        string strSQL = "select tb_News.*,tb_BigClass.Name from tb_News,tb_BigClass where
        tb_News.Flag='已审核' and  tb_News.BigClassID=tb_BigClass.B_id and Info like '%" +
        info + "%' order by tb_News.Cindex";
        return db.ReturnDataSet(strSQL);
    }
    public int GetMaxCindex() //获取最大的 Cindex
    {
        int a = 0;
        string sql = "select * from tb_News";
        if (db.ReturnDataSet(sql).Tables[0].Rows.Count > 0)
        {
            string strSQL = "select max(cindex) as Cindex from tb_News";
            a = int.Parse(db.ReturnDataSet(strSQL).Tables[0].Rows[0][0].ToString());
        }
        return a;
    }
    public DataSet GetNewsOfFlag0() //获取所有未审核的新闻集合
    {
        string strSQL = "select tb_News.*,tb_BigClass.Name from tb_News,tb_BigClass where
        tb_News.Flag='未审核' and tb_News.BigClassID=tb_BigClass.B_id order by  tb_News.
Cindex DESC";
        return db.ReturnDataSet(strSQL);
    }
    public bool CheckNewsByID(int id) //根据 id 审核新闻,审核成功返回 true,否则返回 false
    {

```



```

        string strSQL = "update tb_News set Flag='已审核' where N_id=" + id;
        return db.ExecuteNonQuery(false, strSQL);
    }
    public bool DeleteNewsByBigClassID(int BigClassID) //根据栏目 ID 删除该栏目下的新闻
    {
        string strSQL = "delete from tb_News where BigClassID=" + BigClassID;
        return db.ExecuteNonQuery(false, strSQL);
    }
}
}

```

3. BigClassAccess类实现

程序开发步骤如下：

① 在 DAL 类库中添加新项，选择类，取名为 BigClassAccess。添加后双击“BigClassAccess.cs”文件进行编码。定义所属命名空间为 namespace DAL，类的访问修饰符设为 public。

② 完成主要程序代码。BigClassAccess.cs 类主要通过构造 SQL 语句及调用 DBbase 类中的方法，实现关于新闻类别的显示与检索。其代码如下。

新闻栏目列表的数据访问类：BigClassAccess.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace DAL
{
    #region 数据访问类
    /// <summary>
    /// 新闻栏目的数据访问类
    /// 构造 SQL 语句及调用方法执行
    /// </summary>
    #endregion

    public class BigClassAccess
    {
        DBbase db = new DBbase();
        public int GetMaxCindex() //获取最大的 Cindex
        {
            int a = 0;
            string sql = "select * from tb_BigClass";
            if (db.ReturnDataSet(sql).Tables[0].Rows.Count > 0)
            {
                string strSQL = "select max(Cindex) as Cindex from tb_BigClass";
            }
        }
    }
}

```

```

        a = int.Parse(db.ReturnDataSet(strSQL).Tables[0].Rows[0][0].ToString());
    }
    return a;
}

public bool AddBigClass(string name) //添加新闻栏目,执行成功返回 true,否则返回 false
{
    int cindex = GetMaxCindex() + 1;
    string strSQL = "insert into tb_BigClass (Name,Cindex) values (" + name + "," + cindex + ")";
    return db.ExecuteNonQuery(false, strSQL);
}

public bool DeleteBigClassByID(int id) //根据栏目 id 删除新闻栏目
{
    string strSQL = "delete from tb_BigClass where B_id=" + id;
    return db.ExecuteNonQuery(false, strSQL);
}

public bool UpdateBigClass(Model.BigClassInfo delBC) //根据栏目 id 修改新闻栏目
{
    string strSQL = "update tb_BigClass set Name=" + delBC.Name + " where B_id = " + delBC.B_id;
    return db.ExecuteNonQuery(false, strSQL);
}

public DataSet GetData_BigClass() //执行 SQL 语句,返回整个表里的数据集合
{
    string strSQL = "select * from tb_BigClass";
    return db.ReturnDataSet(strSQL);
}

public DataSet GetBigClass() //获取允许显示的栏目名称
{
    string strSQL = "select B_id,Name from tb_BigClass where Flag='显示' order by Cindex";
    return db.ReturnDataSet(strSQL);
}

public bool UpdateBigClassFlag(string flag, int id) //根据栏目 id 修改栏目的显示状态
{
    string strSQL = "update tb_BigClass set Flag = " + flag + " where B_id=" + id;
    return db.ExecuteNonQuery(false, strSQL);
}

//根据栏目 id 修改栏目名称及显示状态
public bool UpdateBigClassNameAndFlag(int id, string name, string flag)
{
    string strSQL = "update tb_BigClass set Name = " + name + ", Flag = " + flag + " where B_id=" + id;
    return db.ExecuteNonQuery(false, strSQL);
}

public DataSet GetBigClassByID(int M_bc) //根据栏目 id 查询栏目的名称
{
    try
    {

```

```

        string strSQL = "select B_id,Name from tb_BigClass where B_id=" + M_bc;
        return db.ReturnDataSet(strSQL);
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
}

public DataSet GetNewsCount()//获取每个栏目的新闻总条数
{
    string strSQL = "select NewsCount from tb_BigClass where Flag='显示'";
    return db.ReturnDataSet(strSQL);
}

public bool UpdateNewsCountDEL(int BigClassID) //根据所删除的新闻更新栏目下的新闻条数
{
    int newscount = GetNewsCount(BigClassID) - 1;
    string strSQL = "update tb_BigClass set NewsCount=" + newscount + " where B_id=" + BigClassID;
    return db.ExecuteNonQuery(false, strSQL);
}

public bool UpdateNewsCount(int BigClassID) //根据所审核的新闻更新栏目下的新闻条数
{
    int newscount = GetNewsCount(BigClassID) + 1;
    string strSQL = "update tb_BigClass set NewsCount=" + newscount + " where B_id=" + BigClassID;
    return db.ExecuteNonQuery(false, strSQL);
}

public int GetNewsCount(int id) //根据新闻栏目 ID 获取该栏目的新闻数量
{
    string strSQL = "select NewsCount from tb_BigClass where B_id=" + id;
    return int.Parse(db.ReturnDataSet(strSQL).Tables[0].Rows[0][0].ToString());
}
}
}

```

4. FormatString类的实现

程序开发步骤如下：

- ① 在 DAL 类库中添加新项，选择类，取名为 FormatString。添加后双击“FormatString.cs”文件进行编码。定义所属命名空间为 namespace DAL，类的访问修饰符设为 public。
 - ② 完成主要程序代码。FormatString.cs 类主要实现截取定长字符串的操作。其代码如下。
- 字符串处理类：FormatString.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```

namespace DAL
{
    /// <summary>
    /// 字符串处理类
    /// </summary>
    public class FormatString
    {
        #region 字符串截取函数
        /// <summary>
        /// 截取字符串方法
        /// </summary>
        /// <param name="inputString">要截取的原字符串</param>
        /// <param name="len">要截取的长度</param>
        /// <returns>截取后的字符串</returns>
        public string CutString(string str1,int length)
        {
            int s = str1.Length;
            if (length<=s)
            {
                str1 = str1.Substring(0, length)+"...";
            }
            return str1;
        }
        #endregion
    }
}

```

2.5.6 BLL层：业务逻辑类实现

1. NewsLogic类实现

程序开发步骤如下：

① 在 BLL 类库中添加新项，选择类，取名为 NewsLogic。添加后双击“NewsLogic.cs”文件进行编码。定义所属命名空间为 namespace BLL，类的访问修饰符设为 public。

② 完成主要程序代码。NewsLogic.cs 类主要通过对 DAL 类库中 NewsAccess 类方法的逻辑调用，从而实现关于新闻信息的显示与检索。其代码如下。

新闻列表的业务逻辑类：NewsLogic.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace BLL
{

```

```

public class NewsLogic
{
    #region 业务逻辑类
    /// <summary>
    /// 业务逻辑类,统一管理所有的方法和实现页面的调用等!
    /// </summary>
    #endregion
    // 实例化业务逻辑层 DAL 类 NewsAccess 的对象
    DAL.NewsAccess DAL_news = new DAL.NewsAccess();
    // 实例化实体层 Model 类 NewsInfo 的对象
    Model.NewsInfo M_news = new Model.NewsInfo();
    public DataSet GetNewsID()//获取现有新闻 ID
    {
        return DAL_news.GetNewsID();
    }
    public DataSet GetBigClassIDByNewsID(int newsId) //获取现有新闻栏目 ID
    {
        return DAL_news.GetBigClassIDByNewsID(newsId);
    }
    public DataSet GetData_news()//获取全部新闻内容,返回 DataSet 集合
    {
        return DAL_news.GetData_news();
    }
    public DataSet GetData_news(int BigClassID) //获取全部新闻内容
    {
        return DAL_news.GetData_news(BigClassID);
    }
    public DataSet GetDataByBigClass(int BigClassID) //分栏目获取新闻列表
    {
        return DAL_news.GetDataByBigClass(BigClassID);
    }
    public DataSet GetDataByBigClassTopN(int BigClassID,int n) //获取每个分栏前 n 条新闻
    {
        return DAL_news.GetDataByBigClassTopN(BigClassID,n);
    }
    public DataSet GetDataNewN(int n) //获取最新 n 条新闻
    {
        return DAL_news.GetDataNewN(n);
    }
    public DataSet GetDataTopNHits(int n)//获取点击率最高的 n 条新闻标题
    {
        return DAL_news.GetDataTopNHits(n);
    }
    public bool AddNews(Model.NewsInfo m_news) //添加新闻,添加成功返回 true,否则返回 false
    {
        return DAL_news.AddNews(m_news);
    }
}

```

```

public bool DeleteNewsByID(Model.NewsInfo M) //根据新闻 id 删除新闻
{
    return DAL_news.DeleteNewByID(M.N_id);
}
public void UpdateHits(int id) //更新新闻的点击率
{
    DAL_news.UpdateHits(id);
}
public DataSet DataBindNews(int id) //根据指定 id 绑定指定的新闻内容
{
    return DAL_news.DataBindNews(id);
}
public DataSet QueryByNewsTitle(Model.NewsInfo M_news) //按新闻标题模糊查询新闻列表
{
    return DAL_news.QueryByNewsTitle(M_news.Title);
}
public DataSet AdminQueryByNewsTitle(Model.NewsInfo M_news) //按新闻标题模糊查询新闻
{
    return DAL_news.AdminQueryByNewsTitle(M_news.Title);
}
public DataSet QueryByNewsInfo(Model.NewsInfo M_news) //按新闻内容模糊查询新闻
{
    return DAL_news.QueryByNewsInfo(M_news.Info);
}
public DataSet AdminQueryByNewsInfo(Model.NewsInfo M_news) //按新闻内容模糊查询新闻
{
    return DAL_news.AdminQueryByNewsInfo(M_news.Info);
}
public bool UpdateNews(Model.NewsInfo M_news) //修改指定的新闻
{
    return DAL_news.UpdateNews(M_news);
}
public DataSet GetNewsOfFlag0() //获取所有未审核的新闻集合
{
    return DAL_news.GetNewsOfFlag0();
}
public bool CheckNewsByID(Model.NewsInfo M) //根据新闻 id 审核新闻
{
    return DAL_news.CheckNewsByID(M.N_id);
}
//根据栏目 ID 删除该栏目的所有新闻信息
public bool DeleteNewsByBigClassID(Model.NewsInfo Mn)
{
    return DAL_news.DeleteNewsByBigClassID(Mn.BigClassID);
}
}
}

```

2. BigClassLogic类实现

程序开发步骤如下：

① 在 BLL 类库中添加新项，选择类，取名为 BigClassLogic。添加后双击 “BigClassLogic.cs” 文件进行编码。定义所属命名空间为 namespace BLL，类的访问修饰符设为 public。

② 完成主要程序代码。NewsLogic.cs 类主要通过对 DAL 类库中 BigClassAccess 类方法的逻辑调用，从而实现关于新闻栏目的显示与检索。其代码如下。

新闻栏目列表的业务逻辑类：NewsLogic.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace BLL
{
    public class BigClassLogic
    {
        #region 业务逻辑类
        /// <summary>
        /// 业务逻辑类,统一管理所有的方法和实现页面的调用等!
        /// </summary>
        #endregion
        // 实例化业务逻辑层 DAL 类 BigClassAccess 的对象
        DAL.BigClassAccess DAL_BC = new DAL.BigClassAccess();
        // 实例化实体层 Model 类 BigClassInfo 的对象
        Model.BigClassInfo M_BC = new Model.BigClassInfo();
        public bool AddBigClass(Model.BigClassInfo Mb) //添加新闻栏目
        {
            return DAL_BC.AddBigClass(Mb.Name);
        }
        public bool DeleteBigClassByID(Model.BigClassInfo Mb) //根据 id 删除新闻栏目
        {
            return DAL_BC.DeleteBigClassByID(Mb.B_id);
        }
        public bool UpdateBigClass()//修改新闻栏目
        {
            return DAL_BC.UpdateBigClass(M_BC);
        }
        public DataSet GetData_BigClass()//获得 tb_BigClass 表的数据集合
        {
            return DAL_BC.GetData_BigClass();
        }
        public DataSet GetBigClass()//获取允许显示的栏目名称
        {
```

```

        return DAL_BC.GetBigClass();
    }
    //根据 id 修改栏目的显示状态,1 显示,0 不显示
    public bool UpdateBigClassFlag(Model.BigClassInfo Mb)
    {
        return DAL_BC.UpdateBigClassFlag(Mb.Flag, Mb.B_id);
    }
    //根据 id 修改栏目名称及显示状态
    public bool UpdateBigClassNameAndFlag(Model.BigClassInfo Mb)
    {
        return DAL_BC.UpdateBigClassNameAndFlag(Mb.B_id, Mb.Name, Mb.Flag);
    }
    public DataSet GetBigClassByID(Model.BigClassInfo Mb) //根据 id 查询栏目的名称
    {
        return DAL_BC.GetBigClassByID(Mb.B_id);
    }
    public DataSet GetBigClassByID(int id) //根据 id 查询栏目的名称
    {
        return DAL_BC.GetBigClassByID(id);
    }
    public DataSet GetNewsCount()//获取每个栏目的新闻总条数
    {
        return DAL_BC.GetNewsCount();
    }
    //根据所审核的新闻更新栏目下的新闻条数
    public bool UpdateNewsCount(Model.BigClassInfo Mb)
    {
        return DAL_BC.UpdateNewsCount(Mb.B_id);
    }
    //根据所审核的新闻更新栏目下的新闻条数
    public bool UpdateNewsCountDEL(Model.BigClassInfo Mb)
    {
        return DAL_BC.UpdateNewsCountDEL(Mb.B_id);
    }
}
}

```

2.5.7 任务小结

本系统采用的是三层架构模式，本节的主要任务是实现新闻的显示与检索。其中，对数据库的操作 DBbase.cs 类是其他类的基础，要掌握对数据库的操作方法；要注意 ADO.NET 对象的应用及特点。

2.5.8 练习题

1. DataReader 与 DataSet 有什么区别？
2. ADO.NET 有哪些对象？

2.6 新闻评论实现

学习目标

■ 掌握三层架构的层次结构及调用

■ 掌握 SQL 命令的使用

■ 掌握类及方法的实现

■ 建议学时：4 学时

2.6.1 任务名称：新闻评论实现

2.6.2 任务描述

新闻评论是本系统的重要功能之一。本系统采用标准的三层架构。在该任务中，Model 类库里的类 CommentsInfo 主要完成对数据库评论表 tb_Comments 中字段的定义；DAL 类库中的类 CommentsAccess 主要完成对新闻评论的各种功能的具体实现；BLL 类库中的类 CommentsLogic 则完成对 DAL 类库中类的逻辑调用。因此，要完成新闻评论这一任务，就要对此任务的功能进行分析，完成各个类库中类的实现。

2.6.3 任务分析

在本系统中采用标准的三层架构，这三层架构是完成系统前台、后台功能的基础。新闻评论的实现需要完成实体层 Model 类库中的 CommentsInfo 类、数据访问层 DAL 类库中的 CommentsAccess 类、逻辑层 BLL 类库中的 CommentsLogic 类。其中，CommentsInfo 类中定义的属性对应评论表 tb_Comments 中的字段；CommentsAccess 类用于实现新闻评论功能的基本方法；CommentsLogic 类则用于完成对 CommentsAccess 类的逻辑调用，从而实现新闻评论的功能。

本任务完成类的顺序如下：

- ① Model 类库中的 CommentsInfo 类；
- ② DAL 类库中的 CommentsAccess 类；
- ③ BLL 类库中的 CommentsLogic 类。

下面按照分析的思路、方法来完成新闻评论的实现。

2.6.4 Model层：CommentsInfo类实现

程序开发步骤如下：

① 在 Model 类库中添加新项，选择类，取名为 CommentsInfo。添加后双击“CommentsInfo.cs”文件进行编码。定义所属命名空间为 namespace Model，类的访问修饰符设为 public。CommentsInfo 类中主要进行属性的设置，各个属性对应数据库 tb_Comments 表中的相应字段。

② 完成主要程序代码。CommentsInfo 类包含 8 个内部变量，其代码如下。

新闻栏目实体类：CommentsInfo

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;

namespace Model
{
    /// <summary>
    /// 实体层——评论列表
    /// 定义成员变量及属性
    /// </summary>
    public class CommentsInfo
    {
        #region 定义类的成员变量
        /// <summary>
        /// 定义类的成员变量
        /// </summary>
        private int _C_id;        //评论 ID
        private string _C_user; //评论人姓名
        private string _C_qq;    //评论人 QQ
        private string _C_email; //评论人邮件
        private string _C_word;  //评论内容
        private string _C_time;  //评论时间
        private int _newID;      //该新闻在 tb_News 中的 N_id 值
        private int _cindex;     //对同一新闻评论的索引值（若同一新闻有两条评论，则其值按
        先后顺序为 1, 2）
        #endregion

        #region 定义类成员变量的属性
        /// <summary>
        /// 定义类成员变量的属性
        /// </summary>
        public int C_id
        {
            get { return _C_id; }
            set { _C_id = value; }
        }
        public string C_user
        {
            get { return _C_user; }
            set { _C_user = value; }
        }
        public string C_qq
        {
            get { return _C_qq; }
            set { _C_qq = value; }
        }
        public string C_email
        {

```

```

        get { return _C_email; }
        set { _C_email = value; }
    }
    public string C_word
    {
        get { return _C_word; }
        set { _C_word = value; }
    }
    public string C_time
    {
        get { return _C_time; }
        set { _C_time = value; }
    }
    public int NewID
    {
        get { return _newID; }
        set { _newID = value; }
    }
    public int Cindex
    {
        get { return _cindex; }
        set { _cindex = value; }
    }
    #endregion
}
}

```

2.6.5 DAL层：CommentsAccess类实现

程序开发步骤如下：

① 在 DAL 类库中添加新项，选择类，取名为 CommentsAccess。添加后双击“CommentsAccess.cs”文件进行编码。定义所属命名空间为 namespace DAL，类的访问修饰符设为 public。

② 完成主要程序代码。CommentsAccess.cs 类主要通过构造 SQL 语句及调用 DBbase 类中的方法，实现关于新闻评论的显示与检索。其代码如下。

新闻列表的数据访问类：CommentsAccess.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace DAL
{
    /// <summary>

```

```

/// 评论列表的数据访问类
/// 构造 SQL 语句及调用方法执行
/// </summary>
public class CommentsAccess
{
    DBbase db = new DBbase();//实例化基类的对象
    public bool DeleteAllByNewsID(int NewsID) //根据新闻 ID 删除该新闻的全部评论
    {
        string strSQL = "delete from tb_Comments where NewsID=" + NewsID;
        return db.ExecuteNonQuery(false, strSQL);
    }
    public bool DeleteNewsCommentsByCommentsID(int C_id) //根据评论 ID 删除该新闻的该条评论
    {
        string strSQL = "delete from tb_Comments where C_id=" + C_id;
        return db.ExecuteNonQuery(false, strSQL);
    }
    public DataSet GetAllAnswer()//获取全部新闻评论
    {
        string strSQL = "select * from tb_Comments";
        return db.ReturnDataSet(strSQL);
    }
    public DataSet GetAnswerByNewsID(int NewsID) //根据新闻 ID 查询该新闻的最新 3 条评论
    {
        string strSQL = "select top 3 * from tb_Comments where NewsID=" + NewsID + " order by Cindex DESC";
        return db.ReturnDataSet(strSQL);
    }
    public DataSet GetALLAnswerByNewsID(int NewsID) //根据新闻 ID 查询该新闻的全部评论
    {
        string strSQL = "select * from tb_Comments where NewsID=" + NewsID;
        return db.ReturnDataSet(strSQL);
    }
    public DataSet GetCindexByNewsID(int NewsID) //根据新闻 ID 获取该新闻的评论条数
    {
        string strSQL = "select max(Cindex) as Cindex from tb_Comments where NewsID=" + NewsID;
        return db.ReturnDataSet(strSQL);
    }
    public bool AddAnswerByNewsID(Model.CommentsInfo Ma) //根据新闻 ID 添加评论
    {
        int cindex = 1;
        if (GetCindexByNewsID(Ma.NewID).Tables[0].Rows[0][0].ToString() == "")
        {
            cindex = 1;
        }
        else

```

```

        {
            cindex = int.Parse(GetCindexByNewsID(Ma.NewID).Tables[0].Rows[0][0]
                .ToString());
            cindex += 1;
        }
        string strSQL = "insert into tb_Comments (C_user,C_qq,C_email,C_word,C_time,
            NewsID,Cindex) values ('" + Ma.C_user + "','" + Ma.C_qq + "','" + Ma.C_email +
            "','" + Ma.C_word + "','" + Ma.C_time + "','" + Ma.NewID + "','" + cindex + "')";
        return db.ExecuteNonQuery(false, strSQL);
    }
}
}

```

2.6.6 BLL层：CommentsLogic类实现

程序开发步骤如下：

① 在 BLL 类库中添加新项，选择类，取名为 CommentsLogic。添加后双击“CommentsLogic.cs”文件进行编码。定义所属命名空间为 namespace BLL，类的访问修饰符设为 public。

② 完成主要程序代码。CommentsLogic.cs 类主要通过对 DAL 类库中 CommentsAccess 类方法的逻辑调用，从而实现关于新闻评论的显示与检索。其代码如下。

新闻评论列表的业务逻辑类：CommentsLogic.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace BLL
{
    public class CommentsLogic
    {
        #region 业务逻辑类
        /// <summary>
        /// 业务逻辑类,统一管理所有的方法和实现页面的调用等!
        /// </summary>
        #endregion

        DAL.CommentsAccess DAL_C = new DAL.CommentsAccess();
        public bool DeleteAllByNewsID(Model.CommentsInfo Ma) //根据新闻 ID 删除该新闻的全部评论
        {
            return DAL_C.DeleteAllByNewsID(Ma.NewID);
        }
        //根据评论 ID 删除该新闻的该条评论
        public bool DeleteCommentsByCommentsID(Model.CommentsInfo Ma)
        {

```

```

        return DAL_C.DeleteNewsCommentsByCommentsID(Ma.C_id);
    }
    public DataSet GetAllAnswer()//获取全部新闻评论
    {
        return DAL_C.GetAllAnswer();
    }
    public DataSet GetCommentsByNewsID(int newsID) //根据新闻 ID 查询该新闻的全部评论
    {
        return DAL_C.GetAnswerByNewsID(newsID);
    }
    public DataSet GetALLCommentsByNewsID(int newsID) //根据新闻 ID 查询该新闻的全部评论
    {
        return DAL_C.GetALLAnswerByNewsID(newsID);
    }
    public bool AddCommentsByNewsID(Model.CommentsInfo Ma) //根据新闻 ID 添加评论
    {
        return DAL_C.AddAnswerByNewsID(Ma);
    }
    public DataSet GetCindexByNewsID(int NewsID) //根据新闻 ID 获取该新闻的评论条数
    {
        return DAL_C.GetCindexByNewsID(NewsID);
    }
}
}

```

2.6.7 任务小结

本系统采用的是三层架构模式，本节的主要任务是实现新闻评论功能。通过该任务的学习，应加深对数据库操作的认识和理解。

2.7 后台用户管理实现

学习目标

- 掌握三层架构的层次结构及调用
- 掌握 SQL 命令的使用
- 掌握类及方法的实现
- 建议学时：4 学时

2.7.1 任务名称：后台用户管理实现

2.7.2 任务描述

后台用户管理实现是本系统的重要功能之一。本系统采用标准的三层架构。在该任务中，Model 类库中的类 UserInfo 主要完成对数据库用户表 tb_User 中字段的定义；DAL 类库中的

类 `UserAccess` 主要完成对用户操作的各种功能的具体实现；`BLL` 类库中的类 `UserLogic` 则完成对 `DAL` 类库中类的逻辑调用。因此，要完成后台用户管理这一任务，就要对此任务的功能进行分析，完成各个类库中类的实现。

2.7.3 任务分析

在本系统中采用标准的三层架构，这三层架构是完成系统前台、后台功能的基础。后台用户管理实现需要完成实体层 `Model` 类库中的 `UserInfo` 类、数据访问层 `DAL` 类库中的 `UserAccess` 类、逻辑层 `BLL` 类库中的 `UserLogic` 类。其中，`UserInfo` 类中定义的属性对应用户表 `tb_User` 中的字段；`UserAccess` 类用于实现用户操作的基本方法；`UserLogic` 类则用于完成对 `UserAccess` 类的逻辑调用，从而实现用户操作的功能。

本任务完成类的顺序如下：

- ① `Model` 类库中的 `UserInfo` 类；
- ② `DAL` 类库中的 `UserAccess` 类；
- ③ `BLL` 类库中的 `UserLogic` 类。

下面按照分析的思路、方法来完成后台用户管理的实现。

2.7.4 Model层：UserInfo类实现

程序开发步骤如下：

① 在 `Model` 类库中添加新项，选择类，取名为 `UserInfo`。添加后双击 “`UserInfo.cs`” 文件进行编码。定义所属命名空间为 `namespace Model`，类的访问修饰符设为 `public`。`UserInfo` 类中主要进行属性的设置，各个属性对应数据库 `tb_User` 表中的相应字段。

② 完成主要程序代码。`UserInfo` 类包含 5 个内部变量，其代码如下。

用户管理实体类：`UserInfo`

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Model
{
    /// <summary>
    /// 实体层——用户表
    /// 定义成员变量及属性
    /// </summary>
    public class UserInfo
    {
        #region 定义类的成员变量
        /// <summary>
        /// 定义类的成员变量
        /// </summary>
        private int _u_id; //用户 ID
        private string _username; //用户姓名
        private string _password; //用户密码
```

```

private string _useremail; //用户邮箱
private string _Lever;    //用户权限（普通用户或管理员）
#endregion

#region 定义类成员变量的属性
/// <summary>
/// 定义类成员变量的属性
/// </summary>
public string UserName
{
    get { return _username; }
    set { _username = value; }
}
public int U_id
{
    get { return _u_id; }
    set { _u_id = value; }
}
public string Password
{
    get { return _password; }
    set { _password = value; }
}
public string UserEmail
{
    get { return _useremail; }
    set { _useremail = value; }
}
public string Lever
{
    get { return _Lever; }
    set { _Lever = value; }
}
#endregion
}
}

```

2.7.5 DAL层：UserAccess类实现

程序开发步骤如下：

① 在 DAL 类库中添加新项，选择类，取名为 UserAccess。添加后双击 “UserAccess.cs” 文件进行编码。定义所属命名空间为 namespace DAL，类的访问修饰符设为 public。

② 完成主要程序代码。UserAccess.cs 类主要通过构造 SQL 语句及调用 DBbase 类中的方法，实现关于用户的管理。其代码如下。

用户列表的数据访问类：UserAccess.cs

```
using System;
```



```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;

namespace DAL
{
    #region 业务逻辑层
    /// <summary>
    /// 业务逻辑层
    /// 构造 SQL 语句及调用方法执行
    /// </summary>
    #endregion
    public class UserAccess
    {
        DBbase db = new DBbase();
        public int AdminLogin(string UserName, string PassWord) //验证管理员登录的方法
        {
            string strSQL = "select * from tb_User where UserName='" + UserName + "' and Password = '" + PassWord + "' and Lever='管理员'";
            return db.ReturnRowCount(strSQL);
        }
        public bool AddUser(Model.UserInfo del_ad) //添加管理员用户
        {
            string strSQL = "insert into tb_User(UserName,Password,Email,Lever) values ('" + del_ad.UserName + "','" + del_ad.Password + "','" + del_ad.UserEmail + "','" + del_ad.Lever + "')";
            return db.ExecuteNonQuery(false, strSQL);
        }
        public bool DeleteAdmin(Model.UserInfo ma) //删除用户信息
        {
            string strSQL = "delete from tb_User where U_id=" + ma.U_id;
            return db.ExecuteNonQuery(false, strSQL);
        }
        public bool UpdateAdminPassword(Model.UserInfo del_ad) //修改管理员密码
        {
            string strSQL = "update tb_User set Password='" + del_ad.Password + "' where U_id=" + del_ad.U_id;
            return db.ExecuteNonQuery(false, strSQL);
        }
        public bool UpdateAdminAleave(Model.UserInfo del_ad) //修改管理员权限
        {
            string strSQL = "update tb_User set Lever='" + del_ad.Lever + "' where U_id=" + del_ad.U_id;
            return db.ExecuteNonQuery(false, strSQL);
        }
        public DataSet GetDataAdmin()//获得 tb_User 表里的数据集合
    }
}

```

```

    {
        string strSQL = "select * from tb_User";
        return db.ReturnDataSet(strSQL);
    }
    public DataSet QueryUserInfoByID(int id) //根据用户 ID 查询相关信息
    {
        string strSQL = "select * from tb_User where U_id=" + id;
        return db.ReturnDataSet(strSQL);
    }
    public DataSet QueryUserInfoByName(string name) //根据用户名查询相关信息
    {
        string strSQL = "select * from tb_User where UserName=" + name + """;
        return db.ReturnDataSet(strSQL);
    }
    public bool UpdateUserInfoByName(string name, string pwd, string email) //根据用户名修改用户数据
    {
        string strSQL = "update tb_User set Password=" + pwd + ",Email=" + email + ""
        where UserName=" + name + """;
        return db.ExecuteNonQuery(false, strSQL);
    }
    public bool UpdateUserInfo(Model.UserInfo ma) //根据用户 ID 更新相关信息
    {
        string strSQL = "update tb_User set UserName=" + ma.UserName + ",Password=" +
        ma.Password + ",Email=" + ma.UserEmail + ",Lever=" + ma.Lever + " where U_id=" +
        ma.U_id;
        return db.ExecuteNonQuery(false, strSQL);
    }
    public int CheckUser(string UserName) //检测用户名是否存在
    {
        string strSQL = "select * from tb_User where UserName=" + UserName + """;
        return db.ReturnRowCount(strSQL);
    }
    public int UserLogin(string UserName, string PassWord) //验证普通用户登录的方法
    {
        string strSQL = "select * from tb_User where UserName=" + UserName + " and
        Password=" + PassWord + " and Lever='普通用户'";
        return db.ReturnRowCount(strSQL);
    }
}
}

```

2.7.6 BLL层：UserLogic类实现

程序开发步骤如下：

① 在 BLL 类库中添加新项，选择类，取名为 UserLogic。添加后双击 “UserLogic.cs” 文件进行编码。定义所属命名空间为 namespace BLL，类的访问修饰符设为 public。

② 完成主要程序代码。UserLogic.cs 类主要通过 DAL 类库中 UserAccess 类方法的逻辑调用，从而实现关于用户管理的功能。其代码如下。

用户列表的业务逻辑类：UserLogic.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace BLL
{
    public class Userlogic
    {
        DAL.UserAccess DALad = new DAL.UserAccess();//实例化业务逻辑层 DAL 层 UserAccess 的对象
        Model.UserInfo M_ad = new Model.UserInfo();//实例化实体层 Model 层 UserInfo 的对象
        public int AdminLogin(Model.UserInfo Ma) //返回记录总数，检验管理员登录的方法
        {
            return DALad.AdminLogin(Ma.UserName, Ma.Password);
        }
        public bool AddUser(Model.UserInfo M_ad) //添加用户
        {
            return DALad.AddUser(M_ad);
        }
        public bool DeleteAdmin(Model.UserInfo ma) //删除用户信息
        {
            return DALad.DeleteAdmin(ma);
        }
        public bool UpdateAdminPassword()//修改管理员密码
        {
            return DALad.UpdateAdminAleave(M_ad);
        }
        public bool UpdateAdminAleave()//修改管理员权限
        {
            return DALad.UpdateAdminAleave(M_ad);
        }
        public DataSet GetDataAdmin()//获取表里的全部数据
        {
            return DALad.GetDataAdmin();
        }
        public DataSet QueryUserInfoByID(int id) //根据用户 ID 查询相关信息
        {
            return DALad.QueryUserInfoByID(id);
        }
        public DataSet QueryUserInfoByName(Model.UserInfo Ma) //根据用户名查询相关信息
```

```

        {
            return DALad.QueryUserInfoByName(Ma.UserName);
        }
        public bool UpdateUserInfoByName(Model.UserInfo Ma) //根据用户名修改用户数据
        {
            return DALad.UpdateUserInfoByName(Ma.UserName, Ma.Password, Ma.UserEmail);
        }
        public bool UpdateUserInfo(Model.UserInfo ma) //根据用户 ID 修改相关信息
        {
            return DALad.UpdateUserInfo(ma);
        }
        public int CheckUser(Model.UserInfo Ma) //检测用户名是否存在
        {
            return DALad.CheckUser(Ma.UserName);
        }
        Public int UserLogin(Model.UserInfo Ma) //检验普通用户登录的方法
        {
            return DALad.UserLogin(Ma.UserName, Ma.Password);
        }
    }
}

```

2.7.7 任务小结

本系统采用的是三层架构模式，本节的主要任务是实现后台用户的管理功能。通过该任务的学习，应进一步加深对数据库操作的认识和理解。

2.8 母版页设计

学习目标

- 掌握 Web 用户控件的制作
- 掌握母版页的制作
- 建议学时：6 学时

2.8.1 任务名称：母版页设计

2.8.2 任务描述

在界面设计中设计母版页可以保证网站的整体风格。在母版页面中需要完成以下功能：

- 用户注册功能；
- 用户修改注册信息功能；
- 注册用户发布新闻功能；
- 新闻搜索功能；
- 各新闻类别中新闻数量的统计功能。

2.8.3 任务分析

在母版页中需要用到功能相对独立且多次使用的控件,因此要制作相应的 Web 用户控件。在本系统中,制作母版页需要完成以下自定义控件的制作:

- ① 页头控件 Top.ascx, 具有导航功能;
- ② 用户登录控件 Login.ascx, 提供用户注册、登录、发布新闻功能;
- ③ 热点新闻控件 HotNews.ascx, 提供 10 条点击率最高的热点新闻及浏览功能;
- ④ 新闻数量统计控件 Count.ascx, 提供各新闻类型包含的新闻数量的统计功能;
- ⑤ 页尾控件 Bottom.ascx, 提供相关的一些注释信息。

2.8.4 任务完成

设计本系统的前台功能模块时,使用了母版页。母版页及自定义控件的相关知识,请参见第 1 章的相关内容。

在设计过程中,将每个页面都包含的页头、页尾、登录、新闻统计及搜索、热点新闻封装到母版页面中。母版页的设计布局如图 2-4 所示。

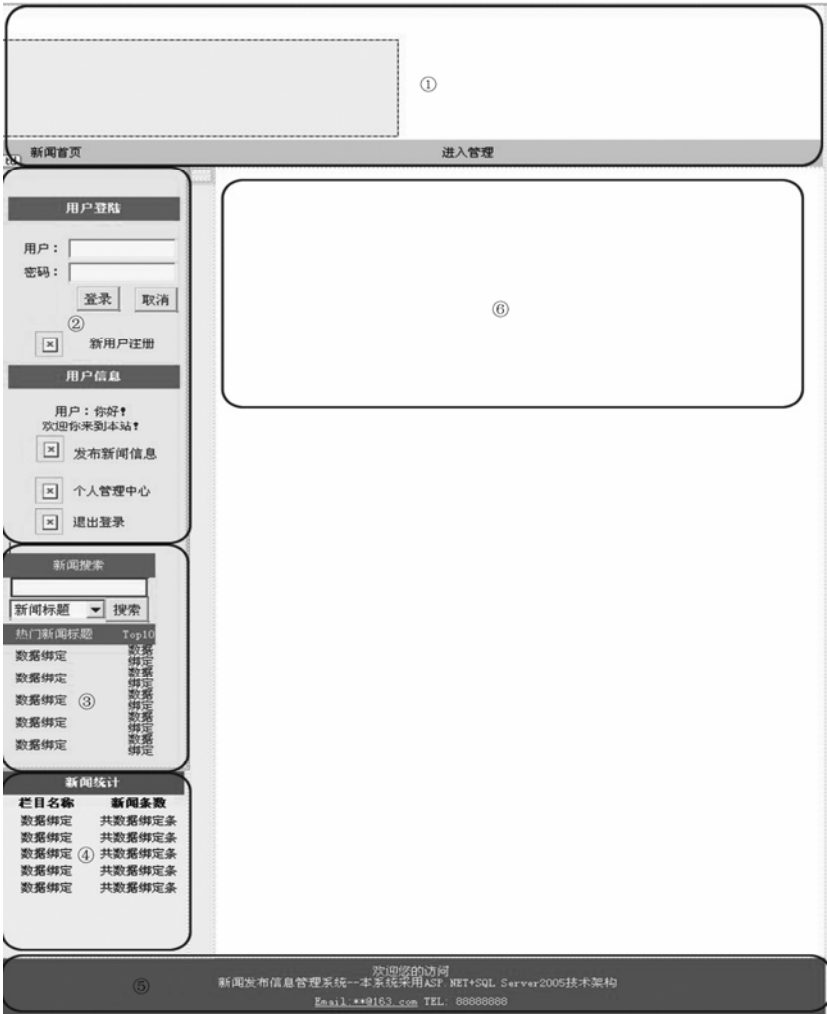


图 2-4 母版页的设计布局

图 2-4 中各部分代表的含义如下:

- ① 页头控件 Top.ascx, 具有导航功能;
- ② 用户登录控件 Login.ascx, 提供用户注册、登录、发布新闻功能;
- ③ 热点新闻控件 HotNews.ascx, 提供 10 条点击率最高的热点新闻及浏览功能;
- ④ 新闻数量统计控件 Count.ascx, 提供各新闻类型包含的新闻数量的统计功能;
- ⑤ 页尾控件 Bottom.ascx, 提供相关的一些注释信息;
- ⑥ 内容页面, 在引用母版页时进行编辑。

建立一个项目, 要先建立解决方案。

① 在已经建立的解决方案上单击鼠标右键, 选择“添加”→“新建网站”并设置网站所在的位置。

② 在新建的网站上单击鼠标右键, 选择“新建文件夹”并命名为 ascx, 用于存放母版页需要的各个控件。

③ 在该文件夹上单击鼠标右键, 选择“添加新项”, 在弹出的窗口中选择“web 用户控件”并将其命名为“Top.ascx”。按此方法依次添加 login.ascx 控件、HotNews.ascx 控件、Count.ascx 控件和 Bottom.ascx 控件。

1. 页头控件Top.ascx

页头控件 Top.ascx 的设计效果如图 2-5 所示, 主要用于实现导航。



图 2-5 页头控件 Top.ascx 的设计效果图

(1) 前台页面设计

源代码中的主要代码如下:

```
<table style="border-style: none; border-color:inherit; border-width: 0px; width:770px;
background-image:url(..\Web/images/top.jpg)" cellpadding="0" cellspacing="0" align="center">
<tr><td style="width: 770px; height:220px;"></td></tr>
<tr align="right"><td id="webjx" style="height:20px"><script type="text/javascript">
setInterval("webjx.innerHTML=new Date().toLocaleString()",1000);</script></td></tr>
<tr><td colspan="2" height="30">
<table style="height:30px; width:100%; border:solid 1px black;" cellpadding="0"
cellspacing="0">
<tr>
<td align="center" bgcolor="" onmouseout="this.bgColor='';" onmouseover="
this.bgColor='#FFFFFF';" style="width: 107px; height: 30px">
<a href="..\web/Default.aspx">
<span style="color: #000000">新闻首页</span></a></td>
<asp:Label ID="sort" runat="server"></asp:Label>
```

```

<td align="center" bgcolor=" " onmouseout="this.bgColor=";" onmouseover="
"this.bgColor='#FFFFFF';" style="height: 30px"><a href=" ../Web/
adminManager/Admin_Login.aspx">进入管理</a></td></tr>

</table>
</td></tr>
</table>

```

(2) 后台功能代码介绍

Top.ascx 控件的主要功能是实现随着新闻类别的变化、相应地动态改变导航栏目。

在编辑器页 (Top.ascx.cs) 编写代码前, 首先需要定义相关类的对象, 以便在编写代码时调用该类中的方法。代码如下:

```
BLL.BigClassLogic B_bc = new BLL.BigClassLogic();
```

程序主要代码如下:

(在 Page_Load 事件中, 通过循环语句读取数据库中允许显示的新闻栏目并显示在导航上)

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        DataSet bigClassNameDS = B_bc.GetBigClass();
        string content = "";
        for (int i = 0; i < bigClassNameDS.Tables[0].DefaultView.Count; i++)
        {
            content = content + "<td align='center' bgcolor=" + " onmouseover=this.bgColor
            = '#FFFFFF'; onmouseout=this.bgColor=" + ">" + "<a href='BigTypeNews.aspx?
            sort=" + bigClassNameDS.Tables[0].Rows[i][0].ToString() + ">" +
            bigClassNameDS.Tables[0].Rows[i][1].ToString() + "</a>
            </td>";
        }
        sort.Text = content;
    }
}

```

2. 用户登录控件Login.ascx

用户登录控件 Login.ascx 的设计效果如图 2-6 所示, 主要用于实现用户登录。

图 2-6 用户登录控件 Login.ascx 的设计效果图

(1) 前台页面设计

首先，将一个表格（Table）置于用户控件（Login.ascx）中，为整个页面进行布局。

然后，从“工具箱”→“标准”选项卡中拖放相应的控件置于表格中，并在各个控件中右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-8 所示。

表 2-8 Login.ascx 中各个控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label	Text 属性设置为 “ ”	显示登录用户的姓名
	Label1	Text 属性设置为 “用户： ”	说明性描述
	Label2	Text 属性设置为 “密码： ”	
	Label3	Text 属性设置为 “用户： ”	
	Label4	Text 属性设置为 “您好！ ”	
	Label5	Text 属性设置为 “欢迎你来到本站！ ”	
	Label6	Text 属性设置为 “用户登录”	
		ForeColor 属性设置为 “White”	
		Font 属性的 Bold 为 “True”	
	Label7	Text 属性设置为 “用户信息”	
ForeColor 属性设置为 “White”			
Font 属性的 Bold 为 “True”			
标准/TextBox 控件	UserName	Text 属性设置为 “ ”	用户输入姓名
	PassWord	Text 属性设置为 “ ” TextMode 属性设置为 “Password”	用户输入密码
标准/LinkButton 控件	AddNews	PostBackUrl 属性设置为 “~/Web/UserAddNews.aspx” Text 属性设置为 “发布新闻信息”	跳转到发布新闻信息页面
	UserCenter	PostBackUrl 属性设置为 “~/Web/UserCenter.aspx”	跳转到个人管理页面
		Text 属性设置为 “个人管理中心”	
	Login_out	Text 属性设置为 “退出登录”	退出登录
标准/Button 控件	Login	Text 属性设置为 “登录”	身份验证，通过后进入欢迎界面
	Cancel	Text 属性设置为 “取消： ”	取消登录

图 2-6 中小手的图标是在源代码中通过 “” 进行添加的。

(2) 后台功能代码介绍

Login.ascx 控件的主要功能是实现用户登录、新闻发布及个人信息的管理。

在编辑器页（Login.ascx.cs）编写代码前，首先需要定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.UserInfo M_userinfo = new Model.UserInfo();//定义 Model 类的对象
BLL.Userlogic B_userlogic = new BLL.Userlogic();//定义 BLL 类的对象
```

程序主要代码如下：


```

static int KKK = 0; // 开关变量
static string KKName = ""; // 存储用户姓名
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (KKK == 0)
        {
            this.login_1.Visible = true; // login_1 用户登录 Table 的 ID
            this.login_2.Visible = false; // login_2 用户信息 Table 的 ID
        }
        else
        {
            this.login_1.Visible = false;
            this.login_2.Visible = true;
            Session["username"] = KKName;
            Label.Text = KKName;
        }
    }
}

protected void login_Click(object sender, EventArgs e) // login_Click() 方法进行密码的验证
{
    M_userinfo.UserName = this.UserName.Text.Trim();
    M_userinfo.Password = FormsAuthentication.HashPasswordForStoringInConfigFile(this.Password.Text.Trim(), "MD5");
    if (UserName.Text == "" || Password.Text == "")
    {
        Response.Write("<script language=javascript>alert('请输入必要信息!');");
        history.back();
    }
    else
    {
        if (UserName.Text != "" && Password.Text != "")
        {
            // 当前用户存在且 Level='管理员'时执行该语句, 跳转到管理页面
            if (B_userlogic.AdminLogin(M_userinfo) > 0)
            {
                Session["admin"] = M_userinfo.UserName.ToString();
                Response.Redirect("../Web/adminManager/Admin_Index.aspx");
            }
            /* 当前用户存在且 Level='普通用户'时执行该语句, 并将 id=login_1 的 table
            设置为 login_1.Visible = false; login_2.Visible = true */
            else if (B_userlogic.UserLogin(M_userinfo) > 0)
            {
                this.login_1.Visible = false;
                this.login_2.Visible = true;
                KKK = 1;
            }
        }
    }
}

```

```

        KKName = UserName.Text.ToString();
        this.Label.Text = KKName;
        Session["username"] = KKName;
    }
    else
    {
        Response.Write("<script language=javascript>alert
        (账号错误! ');history.back();</script>");
    }
    }
}
}
protected void Cancel_Click(object sender, EventArgs e)//Cancel_Click()方法清空已有的信息
{
    this.UserName.Text = "";
    this.PassWord.Text = "";
    KKK = 0;
    this.UserName.Focus();
    Session.Clear();
}

protected void Login_out_Click(object sender, EventArgs e) //退出登录时的设置
{
    this.login_1.Visible = true;
    this.login_2.Visible = false;
    KKK = 0;
    this.UserName.Text = "";
    this.PassWord.Text = "";
    KKName = "";
    this.UserName.Focus();
    Session.Clear();
}
}

```

3. 热点新闻控件HotNews.ascx

热点新闻控件 HotNews.ascx 的设计效果如图 2-7 所示，主要用于实现热点新闻的查询。



图 2-7 热点新闻控件 HotNews.ascx 的设计效果图

(1) 前台页面设计

首先，将一个表格（Table）置于用户控件（HotNews.ascx）中，为整个页面进行布局。

然后，从“工具箱”→“标准”选项卡中拖放相应的控件置于表格中，并在各个控件中右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-9 所示。

表 2-9 HotNews.ascx 中各个控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为 “新闻搜索”	说明性描述
		ForeColor 属性设置为 “White”	
		Font 属性的 Bold 为 “True”	
	Label2	Text 属性设置为 “热门新闻—TOP10”	
		ForeColor 属性设置为 “White”	
		Font 属性的 Bold 为 “True”	
	Label3	Text 属性设置为 “点击率”	
		ForeColor 属性设置为 “White”	
		Font 属性的 Bold 为 “True”	
标准/TextBox 控件	tb_value	Text 属性设置为 “ ”	输入搜索信息
		BorderColor 属性为 “#804000”	
标准/DropDownList 控件	DL_Type	Items 属性设置成员为 “新闻标题”，“新闻标题” 的 Value 属性设置为 “title”，Selected 属性设置为 “True” Items 属性设置成员为 “新闻内容”，“新闻内容” 的 Value 属性设置为 “newinfo”	选择搜索的条件
标准/Button 控件	Search	Text 属性设置为 “搜索”	进行新闻搜索
标准/Repeater 控件	RepHits	需要在源视图下进行设计	用以显示点击率最高的前 10 条新闻名称及访问次数

将 Repeater 控件拖入 HotNews.ascx 的相应位置后，需要切换到源视图进行设置，使 Repeater 控件绑定点击率最高的新闻标题及对应的点击次数。相应的源代码如下：

[illegible]

```
  |
```

(2) 后台功能代码介绍

HotNews.ascx 控件的主要功能是实现热点新闻的查询。

在编辑器页 (HotNews.ascx.cs) 编写代码前, 首先需要定义相关类的对象, 以便在编写代码时调用该类中的方法。代码如下:

```

BLL.NewsLogic B_news = new BLL.NewsLogic();
AL.FormatString D_fstring = new DAL.FormatString();

```

程序主要代码如下:

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        this.RepHits.DataSource = B_news.GetDataTopNHits(10);
        this.RepHits.DataBind();
    }
}
protected void Search_Click(object sender, EventArgs e)
{
    if (this.tb_value.Text.Equals(""))
    {
        Response.Write("<script language=javascript>alert('请输入关键字! ');</script>");
    }
    else
    {
        Response.Redirect("Search.aspx?key=" + tb_value.Text.ToString() + "&type=" +
            DL_Type.SelectedValue.ToString());
    }
}
//截取字符串方法
public string cutString(string str, int len)
{
    return D_fstring.CutString(str, len);
}

```

4. 新闻数量统计控件Count.ascx

新闻数量统计控件 Count.ascx 的设计效果如图 2-8 所示, 主要用于实现各类型新闻数量的统计。

新闻统计	
栏目名称	新闻条数
数据绑定	共数据绑定条
数据绑定	共数据绑定条
数据绑定	共数据绑定条
数据绑定	共数据绑定条
数据绑定	共数据绑定条

图 2-8 新闻数量统计控件 Count.ascx 的设计效果图

(1) 前台页面设计

首先，将一个表格（Table）置于用户控件（Count.ascx）中，为整个页面进行布局。

然后，从“工具箱”→“标准”选项卡中拖放相应的控件置于表格中，并在各个控件中右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-10 所示。

表 2-10 Count.ascx 中各个控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“新闻统计”	说明性描述
		ForeColor 属性设置为“White”	
		Font 属性的 Bold 为“True”	
	Label2	Text 属性设置为“栏目名称”	
		Font 属性的 Bold 为“True”	
	Label3	Text 属性设置为“新闻数量”	
		Font 属性的 Bold 为“True”	
标准/Repeater 控件	Repeater1	需要在源视图下进行设计	显示新闻类型的名称
	Repeater2	需要在源视图下进行设计	显示各新闻类型对应的新闻数量

分别将两个 Repeater 控件拖入 Count.ascx 的相应位置后，需要切换到源视图进行设置。Repeater1 控件绑定新闻类型的名称，相应的源代码如下：

```
<asp:Repeater ID="Repeater1" runat="server">
    <ItemTemplate>
        <table style="width: 82px">
            <tr>
                <td><%#DataBinder.Eval(Container.DataItem,"Name")%></td>
            </tr>
        </table>
    </ItemTemplate>
</asp:Repeater>
```

Repeater2 控件绑定各新闻类型中新闻的数量，相应的源代码如下：

```
<asp:Repeater ID="Repeater2" runat="server">
    <ItemTemplate>
        <table style="width: 82px">
            <tr>
```

```

        <td>共<%#DataBinder.Eval(Container.DataItem,"NewsCount")%>条</td>
    </tr>
</table>
</ItemTemplate>
</asp:Repeater>

```

（2）后台功能代码介绍

Count.ascx 控件的主要功能是实现各个新闻类型新闻数量的统计。

在编辑器页（Count.ascx.cs）编写代码前，首先需要定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
BLL.BigClassLogic B_bigclass = new BLL.BigClassLogic();
```

程序主要代码如下：

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        GetBigClass();
        GetNewsCount();
    }
}
//获取栏目名称
public void GetBigClass()
{
    this.Repeater1.DataSource = B_bigclass.GetBigClass();
    this.Repeater1.DataBind();
}
//获取栏目下的新闻总条数
public void GetNewsCount()
{
    this.Repeater2.DataSource = B_bigclass.GetNewsCount();
    this.Repeater2.DataBind();
}

```

5. 页尾控件Bottom.ascx

页尾控件 Bottom.ascx 的设计效果如图 2-9 所示，主要用于显示一些信息。

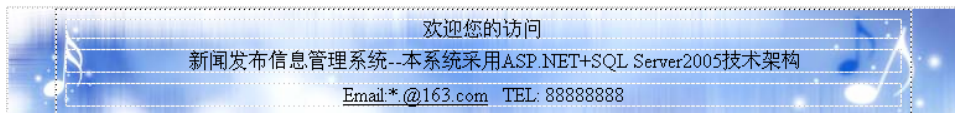


图 2-9 页尾控件 Bottom.ascx 的设计效果图

6. 完成母版页制作

在制作完成各个用户控件后，按照母版页的设计布局图示可以创建母版页。

- ① 在解决方案下，右击网站名称，然后在弹出的快捷菜单中选择“添加新项”命令。
- ② 打开“添加新项”对话框，选择“母版页”并为其命名，然后单击“添加”按钮，即

可创建一个母版页。

③ 设计母版页的布局。在母版页的源视图下为母版页布局，然后从用户控件中将页头控件（Top.ascx）、用户登录控件（Login.ascx）、热点新闻控件（HotNews.ascx）、新闻类型及对应的新闻数量统计控件（Count.ascx）、页尾控件（Bottom.ascx）添加到母版页中。

MasterPage.master 源视图下的主要代码如下：

```
<body>
  <form id="form1" runat="server">
    <div id="total" style="background-color:White; width:770px; height:auto; margin:5px auto; ">
    <div id="top" style=" height:auto; width:100%;">
      <uc1:top ID="top1" runat="server" />
    </div>
    <div id="left" style="width:164px; float:left; height:auto;">
      <uc4:Login ID="Login1" runat="server" />
      <uc2:Left ID="Left1" runat="server" />
      <uc5:Count ID="Count1" runat="server" />
    </div>
    <div id="right" style=" width:auto; float:left;">
      <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
      </div>
    <div id="footer" style=" height:auto; clear:both;">
      <uc3:bottom ID="bottom1" runat="server" />
    </div>
  </div>
</form>
</body>
```

2.8.5 任务小结

通过完成本次任务，应该明确页面的布局方法，以及自定义控件母版页的制作。在自定义控件中，需要对三层架构中的功能进行调用，应掌握其调用方法。

2.8.6 练习题

1. 制作母版页有什么意义？
2. 制作自定义 Web 控件有哪些好处？

2.9 网站前台的实现

学习目标

- 掌握数据控件的用法
- 掌握页面参数的传递及获取
- 掌握三层架构中系统功能的调用
- 建议学时：14 学时

2.9.1 任务名称：网站前台的实现

2.9.2 任务描述

网站前台的实现包括界面和功能的实现。

在前台，用户可以按照新闻类别浏览新闻，同时对新闻进行评论，还可以对需要查找的新闻进行搜索。如果用户想发布自己的新闻内容，需要在前台进行注册，注册以后可以修改自己的注册信息，同时具有发布新闻的权利。当用户发布的新闻被后台的管理员审核通过后，其他用户就可以对该新闻进行浏览及评论。在新闻首页，用户可以看到热点新闻的排名及各个新闻类别中新闻数量的统计。

网站前台的实现包括前台界面的实现和相关功能代码的实现。网站前台主要包括的功能有：

- 用户注册、修改已注册用户信息功能；
- 注册用户发布新闻功能；
- 新闻搜索功能；
- 各新闻类别中新闻数量的统计功能；
- 用户对新闻进行评论的功能；
- 热点新闻统计及浏览功能；
- 按类别浏览新闻功能。

2.9.3 任务分析

网站前台功能的设计主要是数据访问层类、业务逻辑层类及表示层的各功能页，这三层都可以调用实体类。

在实现网站前台功能时要解决好以下 3 个主要问题。

1. 实现表示层的页面对其他层功能的引用

本系统在数据层实现对新闻、评论、用户、新闻类别的基本操作方法，在逻辑层完成对这些方法的逻辑调用，同时会根据各个页面的具体情况增加适当的方法。

因此，在各个功能页面中都离不开对逻辑层相应类的对象的定义及使用。所以要根据不同的页面功能定义相应的对象，从而调用其方法实现功能。

2. 数据控件的应用

本系统中需要对数据进行操作，大量使用了 Repeater 控件。Repeater 控件中的嵌套需要调用 ItemDataBound()方法，通过该方法的实现获取希望得到的数据并实现嵌套的 Repeater 控件的数据绑定。

3. 页面参数的传递

在实现页面功能时，有些页面需要通过参数的传递来获取信息、实现页面功能。因此，如何传递参数并在页面中获取传入的参数是非常重要的。本系统中主要通过 URL 链接地址传递参数，通过 Request.QueryString[参数名称]进行参数的接收。

2.9.4 网站前台首页Default.aspx

双击打开网站下的 Default.aspx 文件。切换到源视图下，在<head>、</head>标记间添加如下源代码：


```
<head runat="server">
    <title>新闻管理系统</title>
    <meta http-equiv=refresh content="1;URL=web/Default.aspx">
</head>
```

这样将网站前台首页定位到该网站下 Web 文件夹下的 Default.aspx 页。

在网站上右击选择“新建文件夹”并命名为“web”，在该文件夹下将存放网站前台的各个功能页面。

在“web”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Default.aspx”，同时勾选“选择母版页”。单击“添加”按钮，选择已编辑完的母版页 MasterPage.master 进行添加。

添加后，双击打开“web”文件夹下的 Default.aspx 页面，进行页面设计。模板页面中的内容页效果如图 2-10 所示。



图 2-10 Default.aspx 内容页面效果图

1. 相关技能、知识

在本系统中，许多页面使用了 Repeater Web 服务器控件。Repeater Web 服务器控件是一个基本容器控件，它可以从页的任何可用数据中创建出自定义列表。其主要作用是循环显示数据库里的数据信息。

添加数据的方式是通过“数据绑定”，然后通过模板来循环显示绑定的数据。所谓模板，就是预先定义的数据显示方式。Repeater 控件允许用户定义如下几种模板。

- HeaderTemplate: 头模板，可选模板。表示 Repeater 控件列表头的内容和布局。
- FooterTemplate: 尾模板，可选模板。表示 Repeater 控件列表尾的内容和布局，如添加一些标注等。
- ItemTemplate: 数据模板，这是 Repeater 数据控件必需的。表示 Repeater 控件每个列表项及其布局。
- AlternatingItemTemplate: 隔行数据模板，可选模板。该模板可以与 ItemTemplate 模板交替使用，以增强 Repeater 控件的显示功能。
- SeparatorTemplate: 分割线模板，可选模板，用于控制每个项目之间显示分割线。

使用一个 Repeater Web 服务器控件需要以下几个步骤。

① 向 ASP.NET 页添加 Repeater Web 服务器控件。

② 将数据源绑定到该Repeater控件上。可以应用数据源控件如 SqlDataSource，或者通过在该页面的编码页面（即.cs页面）进行设置，即通过Repeater的DataSource属性指定数据源，用DataBind()方法进行数据源的绑定。

③ 在该页面前台页面即.aspx页的源视图下，为Repeater控件添加 ItemTemplate模板，该模板应包含数据绑定控件，以便在运行时呈现控件。使用 Eval 数据绑定函数将子控件绑定到指定的数据源，如下面的示例所示用以显示绑定的数据源的值，其中Name是要显示的数据源中列的名称：

```
<%# DataBinder.Eval(Container.DataItem,"Name")%>
```

2. 前台页面设计

网站前台首页（Default.aspx）是母版页的内容页，其设计的具体步骤如下：

首先，将一个表格（Table）控件置于 Default.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应的控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-11 所示。

表 2-11 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“最新新闻” Font 属性的 Bold 为“True”	说明性描述
标准/LinkButton 控件	LinkButton1	Text 属性设置为“更多. . .” PostBackUrl 属性设置为“AllNews.aspx”	链接到 AllNews.aspx 页面， 显示全部新闻
标准/Repeater 控件	RepeaterNew	需要在源视图下进行设计	显示最新的 8 条新闻
标准/Repeater 控件	Repeater1	需要在源视图下进行设计	显示各新闻类别对应的最新 4 条新闻

将 Repeater（控件 ID 名为“RepeaterNew”）控件拖入 Default.aspx 相应位置后，切换到源视图进行设置，使 Repeater 控件绑定点击率最高的新闻标题及对应的点击次数。相应源码的主要代码如下：

```
<asp:Repeater ID="RepeaterNew" runat="server">
  <ItemTemplate>
    <table width="100%" border="0" cellpadding="0" cellspacing="0">
      <tr>
        <td height="24" style="BORDER-bottom: #999999 1px dotted">
          &nbsp;&nbsp;&nbsp;©&nbsp;&nbsp;&nbsp;<span style="font-size:9pt;line-height:15pt">
            <a href="ListView.aspx?cid=<%#DataBinder.Eval(Container.DataItem,"N_id")
            %>'title="" target="_blank"><%#cutString(DataBinder.Eval(Container
            .DataItem,"Title").ToString(),30)%></a></span>
            <font color="#999999"><[<%#Convert.ToDateTime(DataBinder.Eval
            (Container.DataItem,"InfoTime")).ToShortDateString()%></font>
            <font color="#999999">(阅读<font color="#FF0000"><%#DataBinder.Eval
            (Container.DataItem,"Hit")%></font>次)</font>
```

```

        </td>
      </tr>
    </table>
  </ItemTemplate>
</asp:Repeater>

```

将 Repeater（控件 ID 名为“Repeater1”）控件拖入页面相应位置后，切换到源视图进行设置，使 Repeater 控件绑定到新闻标题、新闻发布时间及阅读次数。相应的源代码如下：

```

<asp:Repeater ID="Repeater1" runat="server" OnItemDataBound="Repeater1_ItemDataBound">
  <ItemTemplate><tr><td>
    <table width="100%" border="0" cellpadding="0" cellspacing="0">
      <tr><td width="6%" height="30" rowspan="2"></td>
        <td width="76%" height="28">
          <strong><%#DataBinder.Eval(Container.DataItem,"Name")%></strong></td>
          <td width="18%" align="center"><a href='BigTypeNews.aspx?sort=
            <%#DataBinder.Eval(Container.DataItem,"B_id")%>'>更多...
          </a></td></tr>
        <tr>
          <td height="2" colspan="2" bgcolor="#6699cc"></td></tr>
        <tr>
          <td></td>
          <td colspan="2">
            <asp:repeater id="Repeater2" runat="server">
              <ItemTemplate>
                <table width="100%" border="0" cellpadding="0"
                  cellspacing="0"><tr><td height="24" style="BORDER-bottom:
                    #999999 1px dotted">
                  ©&nbsp;<span style="font-size:9pt;line-height:1.5pt">
                    <a href='ListView.aspx?cid=<%#DataBinder.Eval
                      (Container.DataItem,"N_id")%>' title="" target=
                        "_blank"><%#cutString(DataBinder.Eval(Container
                          .DataItem,"Title").ToString(),30)%></a></span>
                    <font color="#999999"> [<%#Convert.ToDateTime
                      (DataBinder.Eval(Container.DataItem,"infotime")).
                        ToShortDateString()%>]</font>
                    <font color="#999999"> (阅读<font color="#FF0000">
                      <%#DataBinder.Eval(Container.DataItem,"Hit")%>
                    </font>次)</font>
                  </td></tr>
                </table>
              </ItemTemplate>
            </asp:repeater>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>

```

```
</td></tr></ItemTemplate>
</asp:Repeater>
```

3. 后台功能代码

在编辑器页 (Default.aspx.cs) 编写代码前, 需要先定义相关类的对象, 以便在编写代码时调用该类中的方法。代码如下:

```
BLL.NewsLogic B_news = new BLL.NewsLogic();
BLL.BigClassLogic B_bc = new BLL.BigClassLogic();
DAL.FormatString FString = new DAL.FormatString()
```

程序主要代码如下。

① 在 Page_Load 事件中:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        DataBindRepeaterNew();
        DataBindRepeaterBigClass();
        this.Title = "新闻系统——本系统采用 ASP.NET 3.5+SQL Server 2005 技术架构! ";
        Session.Clear();
    }
}
```

② 自定义方法 DataBindRepeaterNew() 获取最新新闻, DataBindRepeaterBigClass() 获取允许显示的栏目名称集合, CutString() 用以截取字符串:

```
public void DataBindRepeaterNew()
{
    RepeaterNew.DataSource = B_news.GetDataNewN(8);
    RepeaterNew.DataBind();
}
public void DataBindRepeaterBigClass()
{
    Repeater1.DataSource = B_bc.GetBigClass();
    Repeater1.DataBind();
}
public string cutString(string str, int len)
{
    return FString.CutString(str, len);
}
```

③ 通过 ItemDataBound 事件对 Repeater1 中的 Repeater2 进行数据绑定:

```
protected void Repeater1_ItemDataBound(object sender, RepeaterItemEventArgs e)
{
    int cid = int.Parse(DataBinder.Eval(e.Item.DataItem, "B_id").ToString());
    Repeater Repeater2;
    Repeater2 = (Repeater)e.Item.FindControl("Repeater2");
```

```
Repeater2.DataSource = B_news.GetDataByBigClassTopN(cid,4);
Repeater2.DataBind();
}
```

2.9.5 新闻栏目页面BigTypeNews.aspx

点击导航栏中的新闻栏目或者点击 Default.aspx 页中新闻栏目右侧对应的“更多...”时将出现此页面。

新闻类别页的功能是使用户可以浏览对应新闻栏目下的所有新闻。

在“web”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“BigTypeNews.aspx”，同时勾选“选择母版页”。单击“添加”按钮，选择已编辑完的母版页 MasterPage.master 进行添加。

添加后，双击打开“web”文件夹下的 BigTypeNews.aspx 页面，进行页面设计。母版页中的内容页面效果如图 2-11 所示。

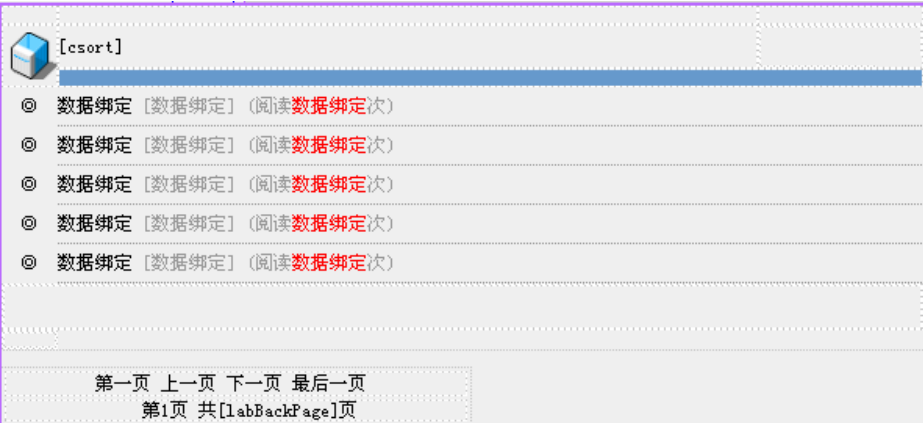


图 2-11 内容页面效果图

1. 相关技能、知识

本系统中的很多页面使用了分页功能。

ASP.NET 提供 3 个功能强大的列表控件：DataGrid、DataList 和 Repeater 控件。相对 DataGrid 控件而言，DataList 和 Repeater 控件具有更高的样式自定义性，但是 Repeater 和 DataList 没有分页功能。使用 PagedDataSource 类可以实现 DataList 和 Repeater 控件的分页显示功能。

PagedDataSource 类封装了 DataGrid 控件的属性，从而使 DataGrid 控件可以执行分页功能。它就是一个数据的容器，先把数据从数据库中读取出来放在这个容器中，然后设置容器的属性取出当前要显示页上的部分数据，最后再将此部分数据绑定到页面上的显示控件中。

PagedDataSource 类的部分公共属性如下。

- AllowCustomPaging：获取或设置指示是否启用自定义分页的值。
- AllowPaging：获取或设置指示是否启用分页的值。
- Count：获取要从数据源使用的项数。
- CurrentPageIndex：获取或设置当前页的索引。
- DataSource：获取或设置数据源。
- DataSourceCount：获取数据源中的项数。

- FirstIndexInPage: 获取页中的第一个索引。
- IsCustomPagingEnabled: 获取一个值，该值指示是否启用自定义分页。
- IsFirstPage: 获取一个值，该值指示当前页是否是首页。
- IsLastPage: 获取一个值，该值指示当前页是否是最后一页。
- IsPagingEnabled: 获取一个值，该值指示是否启用分页。
- IsReadOnly: 获取一个值，该值指示数据源是否是只读的。
- IsSynchronized: 获取一个值，该值指示是否同步对数据源的访问（线程安全）。
- PageCount: 获取显示数据源中的所有项所需要的总页数。
- PageSize: 获取或设置要在单页上显示的项数。
- VirtualCount: 获取或设置在使用自定义分页时数据源中的实际项数。

2. 前台页面设计

新闻栏目页面 BigTypeNews.aspx 是母版页的内容页，其设计的具体步骤如下：

首先，将一个表格（Table）控件置于 BigTypeNews.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应的控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-12 所示。

表 2-12 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	csort	Text 属性设置为 “ ”	显示新闻类别
	Label1	Text 属性设置为 “第”	说明性描述
	labPage	Text 属性设置为 “1”	显示当前页码
	Label2	Text 属性设置为 “页”	说明性描述
	Label3	Text 属性设置为 “共”	说明性描述
	labBackPage	Text 属性设置为 “ ”	显示新闻的总页码
	Label4	Text 属性设置为 “页”	说明性描述
标准/LinkButton 控件	lnkbtnOne	Text 属性设置为 “第一页”	显示第一页新闻
	lnkbtnUp	Text 属性设置为 “上一页”	显示上一页新闻
	lnkbtnNext	Text 属性设置为 “下一页”	显示下一页新闻
	lnkbtnBack	Text 属性设置为 “最后一页”	显示最后一页新闻
标准/Repeater 控件	Repeater1	需要在源视图下进行设计	显示新闻标题、发布时间及阅读次数

将 Repeater（控件 ID 名为 “Repeater1”）控件拖入 BigTypeNews.aspx 相应位置后，切换到源视图进行设置，使 Repeater 控件绑定到新闻标题、新闻发布时间及阅读次数。相应的源代码如下：

```
<asp:Repeater id="Repeater1" runat="server">
  <ItemTemplate>
    <tr>
      <td height="24" align="center">◎</td>
```

```

<td height="24" colspan="2" style="BORDER-BOTTOM: #999999 1px dotted">
    <a href='ListView.aspx?cid=<%#DataBinder.Eval(Container.DataItem,"N_id")%>'
    target="_blank"><%#DataBinder.Eval(Container.DataItem,"Title")%></a>
    <font color="#999999">
    [<%#Convert.ToDateTime(DataBinder.Eval(Container.DataItem,"InfoTime")).
    ToShortDateString()%>]
    (阅读<font color="#ff0000"><%#DataBinder.Eval(Container.DataItem,"Hit")%>
    </font>次)</font>
</td>
</tr>
</ItemTemplate>
</asp:Repeater>

```

3. 后台功能代码

在编辑器页（BigTypeNews.aspx.cs）编写代码前，首先需要定义相关类的对象，以便在编写代码时调用该类中的方法。同时，设置数据成员 `pagesize` 和 `id`。代码如下：

```

BLL.BigClassLogic B_bc = new BLL.BigClassLogic();
Model.BigClassInfo M_bc = new Model.BigClassInfo();
BLL.NewsLogic B_news = new BLL.NewsLogic();
Model.NewsInfo M_news = new Model.NewsInfo();
protected int pagesize;
public int id;

```

程序主要代码如下。

① 在 Page_Load 事件中：

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Request.QueryString["sort"].ToString() != null)
        {
            DataBindcsort();
            M_news.BigClassID = int.Parse(Session["id"].ToString());
            this.Bind();
        }
        else
        {
            Response.Write("<script language=javascript>alert('数据库操作有错误!');");
            Response.Write("window.history.back();</script>");
        }
    }
}

```

② 自定义方法 DataBindcsort(), 动态获取新闻栏目名称：

```

public void DataBindcsort()
{

```

```

Session["id"] = Request.QueryString["sort"].ToString();
M_bc.B_id=Convert.ToInt32(Session["id"].ToString());
if (B_bc.GetBigClassByID(M_bc.B_id).Tables[0].Rows.Count > 0)
{
    this.csort.Text = B_bc.GetBigClassByID(M_bc.B_id).Tables[0].Rows[0][1]
        .ToString();
    this.csort.Font.Bold = true;
    this.Title = B_bc.GetBigClassByID(M_bc.B_id).Tables[0].Rows[0][1].ToString()
        + "专题栏目——本系统采用 ASP.NET+SQL Server 2005 技术架构! ";
}
}
}

```

③ 自定义方法 Bind(), 为新闻分页并绑定新闻数据:

```

public void Bind()
{
    //取得当前页的页码
    int curpage = Convert.ToInt32(this.labPage.Text);
    //使用 PagedDataSource 类实现 Repeater 控件的分页功能
    PagedDataSource ps = new PagedDataSource();
    Session["id"] = Request.QueryString["sort"].ToString();
    M_bc.B_id= Convert.ToInt32(Session["id"].ToString());
    //获取数据集
    DataSet ds = B_news.GetDataByBigClass(M_bc.B_id);
    ps.DataSource = ds.Tables[0].DefaultView;
    //是否可以分页
    ps.AllowPaging = true;
    //显示的数量
    ps.PageSize = 8;
    //取得当前页的页码
    ps.CurrentPageIndex = curpage - 1;
    this.lnkbtnUp.Enabled = true;
    this.lnkbtnNext.Enabled = true;
    this.lnkbtnBack.Enabled = true;
    this.lnkbtnOne.Enabled = true;
    if (ps.IsFirstPage)
    {
        this.lnkbtnOne.Enabled = false;
        this.lnkbtnUp.Enabled = false;
    }
    if (ps.IsLastPage)
    {
        this.lnkbtnNext.Enabled = false;
        this.lnkbtnBack.Enabled = false;
    }
    //显示分页数量
    this.labBackPage.Text = Convert.ToString(ps.PageCount);
}

```



```
//绑定 Repeater1 控件
this.Repeater1.DataSource = ps;
this.Repeater1.DataBind();
}
```

④ 按钮 `lnkbtnOne` 的 Click 事件，执行后跳转到新闻的第一页：

```
protected void lnkbtnOne_Click(object sender, EventArgs e)
{
    this.labPage.Text = "1";
    this.Bind();
}
```

⑤ 按钮 `lnkbtnBack` 的 Click 事件，执行后跳转到新闻的最后一页：

```
protected void lnkbtnBack_Click(object sender, EventArgs e)
{
    this.labPage.Text = this.labBackPage.Text;
    this.Bind();
}
```

⑥ 按钮 `lnkbtnNext` 的 Click 事件，执行后跳转到新闻当前页的下一页：

```
protected void lnkbtnNext_Click(object sender, EventArgs e)
{
    this.labPage.Text = Convert.ToString(Convert.ToInt32(this.labPage.Text) + 1);
    this.Bind();
}
```

⑦ 按钮 `lnkbtnUp` 的 Click 事件，执行后跳转到新闻当前页的上页：

```
protected void lnkbtnUp_Click(object sender, EventArgs e)
{
    this.labPage.Text = Convert.ToString(Convert.ToInt32(this.labPage.Text) - 1);
    this.Bind();
}
```

2.9.6 新闻内容浏览及评论页面 `ListView.aspx`

当点击页面中的新闻标题时会弹出 `ListView.aspx` 页面，该页面的功能是使用户在浏览对应新闻标题下的详细内容时可以对该新闻发表评论。

在“web”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“`ListView.aspx`”，不要勾选“选择母版页”。单击“添加”按钮。

添加后，双击打开“web”文件夹下的 `ListView.aspx` 页面，进行页面设计。页面效果如图 2-12 所示。



图 2-12 ListView.aspx 页面效果图

1. 相关技能、知识

本系统中多次使用了验证控件。ASP.NET 共有 6 种验证控件，分别介绍如下。

(1) RequiredFieldValidator（必须字段验证）

该控件用于验证所要监视的控件是否输入了内容。主要属性：ControlToValidate，表示要进行验证监视的控件；ErrorMessage，表示出错提示信息。使用声明语法格式为：

```
<ASP:RequiredFieldValidator id="Validator_Name" Runat="Server"
    ControlToValidate="要检查的控件名"
    ErrorMessage="出错信息"
    Display="Static|Dymatic|None">
    占位符
```

(2) CompareValidator (比较验证)

该控件用于按设定比较两个输入。使用声明格式如下:

```

<ASP:CompareValidator id="Validator_ID" RunAt="Server"
    ControlToValidate="要验证的控件 ID"
    errorMessage="错误信息"
    ControlToCompare="要比较的控件 ID"
    type="String|Integer|Double|DateTime|Currency"
    operator="Equal|NotEqual|GreaterThan|GreaterTanEqual|LessThan|
    LessThanEqual|DataTypeCheck" Display="Static|Dymatic|None">
    占位符
</ASP:CompareValidator>

```

(3) RangeValidator (范围验证)

该控件用于验证用户输入框输入的内容是否在设定的范围之内。这个控件有 4 个主要属性: **ControlToValidate**, 表示要监视的控件; **MaximumValue**, 表示控制范围的最大值; **MinimumValue**, 表示控制范围的最小值; **ErrorMessage**, 表示当监控的控件输入超出范围时的提示信息。该控件不仅仅限于验证输入的数值, 还可以验证输入的字母。如设定 **MaximumValue=e**, **MinimumValue=a**, 当输入的值在字母顺序表中的 a~e 中时可以接受, 但是, 当输入的值在 f~z 范围内, 就会提示出错。该控件的使用声明格式如下:

```

<ASP:RangeValidator id="Vaidator_ID" Runat="Server"
    controlToValidate="要验证的控件 ID"
    type="Integer"
    MinimumValue="最小值"
    MaximumValue="最大值"
    errorMessage="错误信息"
    Display="Static|Dymatic|None">
    占位符
</ASP:RangeValidator>

```

(4) RegularExpressionValidator (正则表达式验证)

正则表达式验证控件是一个字符串验证控件, 因为结合了 **RegularExpression** (正则表达式), 使其成为 .NET 平台下最强大的字符串验证控件。该控件的主要属性包括: **ControlToValidate**, 表示要进行验证监视的控件; **ErrorMessage**, 表示出错提示信息; **ValidationExpression**, 表示验证表达式。

其使用声明格式如下:

```

<ASP:RegularExpressionValidator id="Validator_ID" RunAt="Server"
    ControlToValidate="要验证控件名"
    ValidationExpression="正则表达式"
    errorMessage="错误信息"
    display="Static">
    占位符
</ASP:RegularExpressionValidator>

```

(5) CustomValidator（自定义验证）

该控件用于用户自定义的函数界定验证方式，用户必须定义一个函数来验证输入。其使用声明格式如下：

```
<ASP:CustomValidator id="Validator_ID" RunAt="Server"
    controlToValidate="要验证的控件"
    onServerValidateFunction="验证函数" errorMessage="错误信息"
    Display="Static|Dymatic|None">
    占位符
</ASP: CustomValidator >
```

(6) ValidationSummary（验证总结）

该控件用于收集本页所有的验证错误信息，并可以将它们组织以后再显示出来。其使用声明格式如下：

```
<ASP:ValidationSummary id="Validator_ID" RunAt="Server"
    HeaderText="错误信息如下： "
    ShowSummary="True|False"
    DiaplayMode="List|BulletList|SingleParagraph">
    占位符
</ASP: ValidationSummary >
```

2. 前台页面设计

新闻内容浏览及评论页面 ListView.aspx，其页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 ListView.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，其中最上面是自定义 Web 控件 Top.ascx，最下面是自定义 Web 控件 Bottom.ascx。在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-13 所示。

表 2-13 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“网友评论” Font 属性中的 Bold 设置为“True”	描述性说明
	Label2	Text 属性设置为“我要评论” Font 属性中的 Bold 设置为“True”	描述性说明
	Label3	Text 属性设置为“更多评论...” Font 属性中的 Bold 设置为“True”	链接到该新闻的全部评论页面
	Label4	Text 属性设置为“网友名称： ”	描述性说明
	Label5	Text 属性设置为“QQ： ”	描述性说明
	Label6	Text 属性设置为“E-mail： ”	描述性说明
	Label7	Text 属性设置为“评论内容： ”	描述性说明
	Label8	Text 属性设置为“打印本页”	打印当前页面
	Label9	Text 属性设置为“关闭窗口”	关闭当前页面
	Label10	Text 属性设置为“本站发表读者评论，只代表读者个人观点”	描述性说明

续表

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Button 控件	btn_Comment	Text 属性设置为“提交评论”	将评论内容提交到数据库、同时显示在当前页面
数据/Repeater 控件	Repeater1	需要在源视图下进行设计	显示当前新闻的信息
	Repeater2	需要在源视图下进行设计	显示最新的 3 条评论信息
验证/ RequiredFieldValidator 控件	RequiredFieldValidator1	ControlToValidate 属性设置为“TextBox1” ErrorMessage 属性设置为“*网友名称不能为空！” Font 属性设置为“Small”	验证用户名不能为空
	RequiredFieldValidator2	ControlToValidate 属性设置为“TextBox2” ErrorMessage 属性设置为“*联系 QQ 不能为空！” Font 属性设置为“Small”	验证 QQ 不能为空
	RequiredFieldValidator3	ControlToValidate 属性设置为“TextBox3” ErrorMessage 属性设置为“*E-mail 不能为空！” Font 属性设置为“Small”	验证 E-mail 不能为空
	RequiredFieldValidator4	ControlToValidate 属性设置为“TextBox4” ErrorMessage 属性设置为“*评论内容不能为空！” Font 属性设置为“Small”	验证评论内容不能为空
验证/ RegularExpressionValidator 控件	RegularExpressionValidator1	ControlToValidate 属性设置为“TextBox3” ErrorMessage 属性设置为“*E-mail 格式不正确！” Font 属性设置为“Small” ValidationExpression 设置为“\w+([-+.]w+)*@\w+([-.]w+)*\.\w+([-.]w+)*”	验证邮件格式
	RegularExpressionValidator2	ControlToValidate 属性设置为“TextBox2” ErrorMessage 属性设置为“*QQ 号只能为数字！” Font 属性设置为“Small” ValidationExpression 设置为“^[0-9]+\$”	验证 QQ 格式
标准/TextBoxt 控件	TextBox1	Text 属性设置为“ ” MaxLength 设置为“20”	用户在此输入姓名
	TextBox2	Text 属性设置为“ ” MaxLength 设置为“9”	用户在此输入 QQ
	TextBox3	Text 属性设置为“ ” MaxLength 设置为“40”	用户在此输入 E-mail

控 件 类 型	控 件 ID	主要属性设置	用 途
	TextBox4	Text 属性设置为 “ ” MaxLength 设置为 “ 100 ”	用户在此输入 评论内容

将 Repeater（控件 ID 名为 “Repeater1”）控件拖入 ListView.aspx 相应位置后，切换到源视图进行设置，使 Repeater 控件绑定到新闻的相关信息。相应源代码如下：

```
<asp:repeater id="repeater1" Runat="server">
<ItemTemplate>
  <table width="95%" border="0" align="center" cellpadding="0" cellspacing="0">
    <tr>
      <td height="50" colspan="2" align="center" class="tit"><%#DataBinder.
        Eval(Container.DataItem,"title")%></td>
    </tr>
    <tr>
      <td width="40%" height="30" style="BORDER-TOP: #666666 1px solid;
        BORDER-BOTTOM: #666666 1px solid">双击自动滚屏</td>
      <td width="60%" align="center" style="BORDER-TOP: #666666 1px solid;
        BORDER-BOTTOM: #666666 1px solid">发布者： <%#DataBinder.Eval
        (Container.DataItem,"username")%>发布时间： <%#Convert.ToDateTime
        (DataBinder.Eval(Container.DataItem,"InfoTime")).ToShortDateString()%>
        阅读： <font color="#ff0000"><%#DataBinder.Eval(Container.DataItem,
        "Hit")%></font>次 评论： <font color="#ff0000">
        <%#GetAnswerCindexByNewsID(int.Parse(DataBinder.Eval
        (Container.DataItem, "N_id").ToString()))%></font>次</td>
    </tr>
    <tr>
      <td colspan="2"><br><div style=FONT-SIZE:12px'><%#checkcontent
        (DataBinder.Eval(Container.DataItem,"info").ToString())%></div></td>
    </tr>
    <tr align="right">
      <td colspan="2" height="30"></td>
    </tr>
    <tr align="right">
      <td colspan="2"></td>
    </tr>
  </table>
</ItemTemplate>
</asp:repeater>
```

将 Repeater（控件 ID 名为 “Repeater2”）控件拖入 ListView.aspx 相应位置后，切换到源视图进行设置，使 Repeater 控件绑定到该新闻最新的 3 条评论。相应源代码如下：

```
<asp:Repeater ID="Repeater2" runat="server">
<ItemTemplate>
  <table width="95%" border="0" cellspacing="0" cellpadding="0">
```

```

<tr><hr /></tr>
<tr><td colspan="8">
    <table width="100%" border="0" cellspacing="0" cellpadding="0">
        <tr> <td width="55" align="right">网友姓名:</td>
            <td width="60" align="left"><%#DataBinder.Eval(Container.DataItem,
                "C_user")%>
            <td width="55" align="right">留言时间:</td>
            <td width="130" align="left"><%#DataBinder.Eval(Container.DataItem,
                "C_time")%></td>
            <td width="30" align="right">QQ:</td>
            <td width="80" align="left"><%#DataBinder.Eval(Container.DataItem,
                "C_qq")%></td>
            <td width="55" align="right">E-mail:</td>
            <td width="90" align="left"><a href="mailto:<%#DataBinder.Eval(
                Container.DataItem,"C_email")%>"><%#DataBinder.Eval(Container.
                DataItem,"C_email")%></a></td>
            <td width="150" align="right">第<font color="#FF0000">
                <%#DataBinder.Eval(Container.DataItem,"cindex")%></font>楼</td>
        </tr>
    </table>
</td></tr>
<tr><td colspan="8"><hr /></td></tr>
<tr>
    <td width="55" align="left" valign="top">留言内容:</td>
    <td colspan="6"><div align="left" valign="top"><%#DataBinder.Eval
        (Container.DataItem,"C_word")%></div></td>
</tr>
</table>
</ItemTemplate>
</asp:Repeater>

```

Label3（更多评论…）源代码如下：

```

<td align="right" bgcolor="#74ECE8">
    <a href="MoreComments.aspx?NewsID=<%=NewsID()%>&NewsTitle=<%=NewsTitle()%>">
        <asp:Label ID="Label3" runat="server" Font-Bold="True" Text="更多评论...">
        </asp:Label>
    </a>&nbsp;&nbsp;&nbsp;&nbsp;<br />&nbsp;&nbsp;&nbsp;&nbsp;
</td>

```

Label8（打印本页）和 Label9（关闭窗口）源代码如下：

```

<td align="right" style="height: 16px"><a onclick="javascript:window.open
('pinglun.asp?id=',",width=295,height=185,toolbar=no, status=no, menubar=no,
resizable=yes, scrollbars=no');return false;"
href="#"></a> &nbsp;&nbsp;&nbsp;&nbsp;
<a href="javascript:window.print()">

```

```

        <asp:Label ID="Label8" runat="server" Text="打印本页"></asp:Label></a>
        | <IMG height="14" src="images/close.gif" width="14" align="absMiddle">
        <a href="javascript:window.close()">
        <asp:Label ID="Label9" runat="server" Text="关闭窗口"></asp:Label></a>
    </td>

```

在当前窗口双击鼠标时，窗口自动向下滚动，单击时停止滚动。这需要在源视图中的<head>和</head>标记之间添加脚本，源代码如下：

```

<script type="text/javascript">
    var currentpos,timer;
    function initialize(){
        currentpos=0;
        timer=setInterval("scrollPage()",1); //使用定时器不断执行滚动操作
    }
    function stopScroll(){
        clearInterval(timer); //清空页面中的定时器
    }
    function scrollPage(){
        currentpos++;
        window.scroll(0,currentpos); //滚屏操作
    }
    document.onmousedown=stopScroll; //开始滚屏
    document.ondblclick=initialize; //结束滚屏
</script>

```

3. 后台功能代码

在编辑器页（ListView.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```

Model.NewsInfo M_news = new Model.NewsInfo();
BLL.NewsLogic B_news = new BLL.NewsLogic();
Model.CommentsInfo M_comments = new Model.CommentsInfo();
BLL.CommentsLogic B_comments = new BLL.CommentsLogic();

```

程序主要代码如下。

① 在 Page_Load 事件中：

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (Request.QueryString["cid"] != null)
        {
            ViewState["cid"] = Request.QueryString["cid"];
            M_news.N_id = int.Parse(Request.QueryString["cid"].ToString());
            B_news.UpdateHits(M_news.N_id);
            DataBindNews();
            this.Title = NewsTitle();
        }
    }
}

```



```

        M_comments.NewID = M_news.N_id;
        DataBindAnswer(M_comments.NewID);
    }
    else
    {
        Response.Write("<script language=javascript>alert('数据库操作有错误!');");
        Response.Write("window.history.back();</script>");
    }
}
}

```

② 自定义方法 `NewsTitle()`，根据当前新闻 ID 获取新闻的标题：

```

public string NewsTitle()
{
    return B_news.DataBindNews(int.Parse(ViewState["cid"].ToString())).Tables[0].
    Rows[0][1].ToString();
}

```

③ 自定义方法 `GetAnswerCindexByNewsID()`，根据当前新闻 ID 获取该新闻的评论总数：

```

public string GetAnswerCindexByNewsID(int NewsID)
{
    return B_comments.GetCindexByNewsID(NewsID).Tables[0].Rows[0][0].ToString();
}

```

④ 自定义方法 `NewsID()`，获取当前新闻的 ID：

```

public string NewsID()
{
    return ViewState["cid"].ToString();
}

```

⑤ 自定义方法 `DataBindNews()`，给控件绑定新闻的数据源：

```

public void DataBindNews()
{
    DataSet ds = B_news.DataBindNews(M_news.N_id);
    this.repeater1.DataSource = ds;
    this.repeater1.DataBind();
    ds.Clear();
    ds.Dispose();
}

```

⑥ 自定义方法 `DataBindAnswer()`，绑定评论数据到控件（前 3 条评论）：

```

public void DataBindAnswer(int newsID)
{
    DataSet ds = B_comments.GetCommentsByNewsID(newsID);
    this.Repeater2.DataSource = ds;
    this.Repeater2.DataBind();
}

```

```

        ds.Clear();
        ds.Dispose();
    }

```

⑦ 自定义方法 checkcontent(), 信息验证:

```

public string checkcontent(string content)
{
    content = Regex.Replace(content, @"&nbsp;", @" ");
    content = Regex.Replace(content, @"&", @"&");
    content = Regex.Replace(content, @"<", @"<");
    content = Regex.Replace(content, @">", @">");
    content = Regex.Replace(content, @""", @"");
    content = Regex.Replace(content, @"../..\/uppic\/", @"../Web/uppic/");
    return content;
}

```

⑧ btn_Comment 的 Click 事件, 将评论信息添加到数据库中、同时当前页面显示:

```

protected void btn_Comment_Click(object sender, EventArgs e)
{
    M_comments.C_user = this.TextBox1.Text.Trim();
    M_comments.C_qq = this.TextBox2.Text.Trim();
    M_comments.C_email = this.TextBox3.Text.Trim();
    M_comments.C_word = this.TextBox4.Text.Trim();
    M_comments.C_time = System.DateTime.Now.ToString();
    M_comments.NewID = int.Parse(ViewState["cid"].ToString());
    if (B_comments.AddCommentsByNewsID(M_comments))
    {
        Response.Write("<script language=javascript>alert('添加评论成功! ');");
    }
    else
    {
        Response.Write("<script language=javascript>alert('添加评论失败! ');");
    }
    //重载新闻评论信息
    DataBindAnswer(M_comments.NewID);
    //清空文本控件的值
    this.TextBox1.Text = "";
    this.TextBox1.Focus();
    this.TextBox2.Text = "";
    this.TextBox3.Text = "";
    this.TextBox4.Text = "";
}

```

2.9.7 新闻全部评论浏览页面MoreComments.aspx

当单击新闻内容浏览及评论页面 ListView.aspx 页面中的“更多评论...”时，将弹出 MoreComments.aspx 页面。该页面的功能是使用户可以浏览对应新闻的所有新闻评论。

在“web”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“MoreComments.aspx”，不要勾选“选择母版页”。单击“添加”按钮。

添加后，双击打开“web”文件夹下的 MoreComments.aspx 页面，进行页面设计。页面效果如图 2-13 所示。



图 2-13 MoreComments.aspx 页面效果图

1. 前台页面设计

新闻全部评论浏览页面 MoreComments.aspx，其页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 MoreComments.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，其中最上面是自定义 Web 控件 Top.ascx，最下面是自定义 Web 控件 Bottom.ascx。在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-14 所示。

表 2-14 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为 “第”	说明性描述
	labPage	Text 属性设置为 “1”	显示当前页码
	Label2	Text 属性设置为 “页”	说明性描述
	Label3	Text 属性设置为 “共”	说明性描述
	labBackPage	Text 属性设置为 “ ”	显示新闻评论的总页码
	Label4	Text 属性设置为 “页”	说明性描述
标准/LinkButton 控件	lnkbtnOne	Text 属性设置为 “第一页”	显示第一页新闻
	lnkbtnUp	Text 属性设置为 “上一页”	显示上一页新闻
	lnkbtnNext	Text 属性设置为 “下一页”	显示下一页新闻
	lnkbtnBack	Text 属性设置为 “最后一页”	显示最后一页新闻
数据/Repeater 控件	Repeater1	需要在源视图下进行设计	显示新闻评论的信息

将 Repeater（控件 ID 名为 “Repeater1”）控件拖入 MoreComments.aspx 相应位置后，切换到源视图进行设置，使 Repeater 控件绑定到新闻评论的相关信息。相应源代码如下：

```
<asp:Repeater ID="Repeater1" runat="server">
<ItemTemplate>
    <table width="95%" border="0" cellspacing="0" cellpadding="0">
        <tr><hr /></tr>
        <tr><td colspan="8">
            <table width="100%" border="0" cellspacing="0" cellpadding="0">
                <tr>
                    <td width="55" align="center">留言用户:</td><td width="60"
                    align="left"><%#
                    DataBinder.Eval(Container.
                    DataItem,"C_user")%>
                    <td width="55" align="center">留言时间:</td>
                    <td width="60" align="left"><%#Convert.ToDateTime(DataBinder.
                    Eval(Container.DataItem, "C_time")).ToShortDateString()%>
                    </td>
                    <td width="55" align="center">QQ:</td>
                    <td width="60" align="left"><%#DataBinder.Eval(Container.
                    DataItem,"C_qq")%>
                    <td width="55" align="center">E-mail:</td>
                    <td width="90" align="left"><a href="mailto:<%#DataBinder.
                    Eval(Container.DataItem,"C_email")%>"><%#DataBinder.
                    Eval(Container.DataItem,"C_email")%></a></td>
                    <td width="150" align="right">第<font color="#FF0000">
                    <%#DataBinder.Eval(Container.DataItem,"Cindex")%>
                    </font>楼</td>
                </tr>
            </table>
        </td>
    </tr>
</table>
```

```

        </td></tr>
        <tr><td colspan="8"><hr /></td></tr>
        <tr>
            <td width="70" align="left" valign="top">留言内容:</td>
            <td colspan="7"><div align="left" valign="top"><%#DataBinder.
                Eval(Container.DataItem,"C_word")%></div></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td>&nbsp;</td>
        </tr>
    </table>
</ItemTemplate>
</asp:Repeater>

```

2. 后台功能代码

在编辑器页（MoreComments.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```

Model.CommentsInfo M_comments = new Model.CommentsInfo();
BLL.CommentsLogic B_comments = new BLL.CommentsLogic();

```

程序主要代码如下。

① 在 Page_Load 事件中：

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Request.QueryString["NewsID"] == null)
        {
            Response.Write("<script language=javascript>alert(数据库操作出错! )
                </script>");
        }
        else
        {
            this.Bind();
            //设置评论页面的标题
            this.Title = Request.QueryString["NewsTitle"].ToString() + "的全部评论
                列表! ";
        }
    }
}

```

② 自定义方法 Bind()，为新闻评论分页并绑定评论数据：

```

public void Bind()
{
    int curpage = Convert.ToInt32(this.labPage.Text);

```

```

//使用 PagedDataSource 类实现 Repeater 控件的分页功能
PagedDataSource ps = new PagedDataSource();
M_comments.NewID = int.Parse(Request.QueryString["NewsID"].ToString());
DataSet ds =B_comments.GetALLCommentsByNewsID(M_comments.NewID);
ps.DataSource = ds.Tables[0].DefaultView;
ps.AllowPaging = true;
ps.PageSize =6;
ps.CurrentPageIndex = curpage - 1;
this.lnkbtnUp.Enabled = true;
this.lnkbtnNext.Enabled = true;
this.lnkbtnBack.Enabled = true;
this.lnkbtnOne.Enabled = true;
if (ps.IsFirstPage)
{
    this.lnkbtnOne.Enabled = false;
    this.lnkbtnUp.Enabled = false;
}
if (ps.IsLastPage)
{
    this.lnkbtnNext.Enabled = false;
    this.lnkbtnBack.Enabled = false;
}
this.labBackPage.Text = Convert.ToString(ps.PageCount);
this.Repeater1.DataSource = ps;
this.Repeater1.DataBind();
}

```

③ 按钮 lnkbtnOne、lnkbtnBack、lnkbtnNext、lnkbtnUp 的 Click 事件可参见 2.9.5 节内容。

2.9.8 全部新闻页面AllNews.aspx

当点击网站前台首页 Web/Default.aspx 页面中“最新新闻”右侧的“更多”时，将弹出全部新闻页面 AllNews.aspx。该页面的功能是使用户可以浏览全部的新闻内容。

在“web”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“AllNews.aspx”，勾选“选择母版页”。单击“添加”按钮。

添加后，双击打开“web”文件夹下的 AllNews.aspx 页面，进行页面设计。母版页中的内容页面效果如图 2-14 所示。



图 2-14 内容页面效果图

1. 前台页面设计

全部新闻页面 AllNews.aspx 是母版页的内容页，其设计的具体步骤如下：

首先，将一个表格（Table）控件置于 AllNews.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-15 所示。

表 2-15 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为 “第”	说明性描述
	LabPage	Text 属性设置为 “1”	显示当前页码
	Label2	Text 属性设置为 “页”	说明性描述
	Label3	Text 属性设置为 “共”	说明性描述
	labBackPage	Text 属性设置为 “ ”	显示新闻的总页码
	Label4	Text 属性设置为 “页”	说明性描述
标准/LinkButton 控件	lnkbtnOne	Text 属性设置为 “第一页”	显示第一页新闻
	lnkbtnUp	Text 属性设置为 “上一页”	显示上一页新闻
	lnkbtnNext	Text 属性设置为 “下一页”	显示下一页新闻
	lnkbtnBack	Text 属性设置为 “最后一页”	显示最后一页新闻
数据/Repeater 控件	Repeater1	需要在源视图下进行设计	显示新闻标题、发布时间及阅读次数

将 Repeater（控件 ID 名为 “Repeater1”）控件拖入 AllNews.aspx 相应位置后，切换到源视图进行设置，使 Repeater 控件绑定到新闻标题、新闻发布时间及阅读次数。相应的源代码如下：

```
<asp:Repeater ID="Repeater1" runat="server">
  <ItemTemplate>
    <table width="605px" border="0" cellpadding="0" cellspacing="0">
      <tr>
        <td height="24" style="BORDER-bottom: #999999 1px dotted">&nbsp;  ©&nbsp;  
          <span style="font-size:9pt;line-height:15pt"><a href=ListView.aspx?
            cid=< %#DataBinder.Eval(Container.DataItem,"N_id")%>' title=""
            target="_blank">< %#cutString(DataBinder.Eval(Container.DataItem,
              "title").ToString(),30)%></a>
          </span>   <font color="#999999" >[< %#Convert.ToDateTime(DataBinder.Eval
            (Container.DataItem,"InfoTime")).ToShortDateString()%>]</font>
          <font color="#999999" >(阅读<font color="#FF0000">< %#DataBinder.Eval
            (Container.DataItem,"Hit")%></font>次)</font>
        </td>
      </tr>
    </table>
  </ItemTemplate>
</asp:Repeater>
```

2. 后台功能代码

在编辑器页（AllNews.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
BLL.NewsLogic B_news = new BLL.NewsLogic();
DAL.FormatString D_fs = new DAL.FormatString();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        this.Title = "ASP.NET 新闻系统——查看全部新闻信息！";
        this.Bind();
    }
}
```

② 自定义方法 Bind()，为新闻分页并绑定新闻数据：

```
public void Bind()
{
    //取得当前页的页码
    int curpage = Convert.ToInt32(this.labPage.Text);
    //使用 PagedDataSource 类实现 Repeater 控件的分页功能
    PagedDataSource ps = new PagedDataSource();
    DataSet ds = B_news.GetData_news();
    ps.DataSource = ds.Tables[0].DefaultView;
    //是否可以分页
    ps.AllowPaging = true;
    //显示的数量
    ps.PageSize = 20;
    //取得当前页的页码
    ps.CurrentPageIndex = curpage - 1;
    this.lnkbtnUp.Enabled = true;
    this.lnkbtnNext.Enabled = true;
    this.lnkbtnBack.Enabled = true;
    this.lnkbtnOne.Enabled = true;
    if (ps.IsFirstPage)
    {
        this.lnkbtnOne.Enabled = false;
        this.lnkbtnUp.Enabled = false;
    }
    if (ps.IsLastPage)
    {
        this.lnkbtnNext.Enabled = false;
        this.lnkbtnBack.Enabled = false;
    }
}
```



```
}
//显示分页数量
this.labBackPage.Text = Convert.ToString(ps.PageCount);
//绑定 Repeater1 控件
this.Repeater1.DataSource = ps;
this.Repeater1.DataBind();
}
```

③ 按钮 lnkbtnOne、lnkbtnBack、lnkbtnNext、lnkbtnUp 的 Click 事件可参见 2.9.5 节内容。

2.9.9 新闻搜索页面Search.aspx

当在网站前台首页 Web/Default.aspx 页面中进行新闻搜索时，单击“搜索”按钮，会弹出搜索结果页面 Search.aspx。该页面的功能是显示用户搜索到的新闻。

在“web”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Search.aspx”，勾选“选择母版页”。单击“添加”按钮。

添加后，双击打开“web”文件夹下的 Search.aspx 页面，进行页面设计。母版页中的内容页面效果如图 2-15 所示。

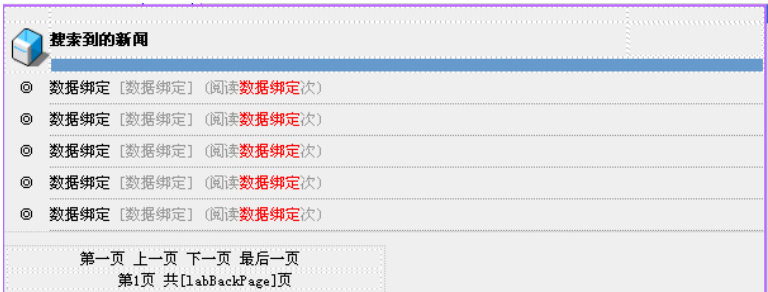


图 2-15 内容页面效果图

1. 前台页面设计

新闻搜索页面 Search.aspx 是母版页的内容页，其设计的具体步骤如下：

首先，将一个表格（Table）控件置于 Search.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-16 所示。

表 2-16 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	SearchNews	Text 属性设置为“搜索到的新闻” Font 属性中的 Bold 设置为“True”	说明性描述
	Label1	Text 属性设置为“第”	说明性描述
	labPage	Text 属性设置为“1”	显示当前页码
	Label2	Text 属性设置为“页”	说明性描述
	Label3	Text 属性设置为“共”	说明性描述
	labBackPage	Text 属性设置为“ ”	显示新闻的总页码
	Label4	Text 属性设置为“页”	说明性描述

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/LinkButton 控件	lnkbtnOne	Text 属性设置为 “第一页”	显示第一页新闻
	lnkbtnUp	Text 属性设置为 “上一页”	显示上一页新闻
	lnkbtnNext	Text 属性设置为 “下一页”	显示下一页新闻
	lnkbtnBack	Text 属性设置为 “最后一页”	显示最后一页新闻
数据/Repeater 控件	Repeater1	需要在源视图下进行设计	显示搜索到的新闻标题、发布时间及阅读次数

将 Repeater（控件 ID 名为 “Repeater1”）控件拖入 Search.aspx 相应位置后，切换到源视图进行设置，使 Repeater 控件绑定到新闻标题、新闻发布时间及阅读次数。相应的源代码如下：

```
<asp:Repeater id="Repeater1" runat="server">
  <ItemTemplate>
    <tr>
      <td height="24" align="center">◎</td>
      <td height="24" colspan="2" style="BORDER-BOTTOM: #999999 1px dotted">
        <a href="ListView.aspx?cid=<%#DataBinder.Eval(Container.DataItem,"N_id")
          %>' target="_blank"><%#DataBinder.Eval(Container.DataItem,"Title")%>
        </a><font color="#999999">[<%#Convert.ToDateTime(DataBinder.Eval
          (Container.DataItem,"Infotime")).ToShortDateString()%>] (阅读
        <font color="#ff0000"><%#DataBinder.Eval(Container.DataItem,"Hit")%>
        </font>>次)</font>
      </td>
    </tr>
  </ItemTemplate>
</asp:Repeater>
```

2. 后台功能代码

在编辑器页（Search.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.NewsInfo M_news = new Model.NewsInfo();
BLL.NewsLogic B_news = new BLL.NewsLogic();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
  if (!Page.IsPostBack)
  {
    this.Bind();
    this.Title = "新闻搜索页面——查看搜索新闻列表！！";
  }
}
```

② 自定义方法 Bind(), 为搜索到的新闻分页并绑定新闻数据:

```
public void Bind()
{
    //取得当前页的页码
    int curpage = Convert.ToInt32(this.labPage.Text);
    //使用 PagedDataSource 类实现 Repeater 控件的分页功能
    PagedDataSource ps = new PagedDataSource();
    //获取查询页面传递过来的查询关键字
    string key = Request.QueryString["key"].ToString();
    //获取查询页面传递过来的查询类别 [新闻标题 | 新闻内容]
    string type = Request.QueryString["type"].ToString();
    DataSet ds;
    if (type.Equals("title"))
    {
        M_news.Title = key;
        ds = B_news.QueryByNewsTitle(M_news);
    }
    else
    {
        M_news.Info = key;
        ds = B_news.QueryByNewsInfo(M_news);
    }
    ps.DataSource = ds.Tables[0].DefaultView;
    //是否可以分页
    ps.AllowPaging = true;
    //显示的数量
    ps.PageSize = 10;
    //取得当前页的页码
    ps.CurrentPageIndex = curpage - 1;
    this.lnkbtnUp.Enabled = true;
    this.lnkbtnNext.Enabled = true;
    this.lnkbtnBack.Enabled = true;
    this.lnkbtnOne.Enabled = true;
    if (ps.IsFirstPage)
    {
        this.lnkbtnOne.Enabled = false;
        this.lnkbtnUp.Enabled = false;
    }
    if (ps.IsLastPage)
    {
        this.lnkbtnNext.Enabled = false;
        this.lnkbtnBack.Enabled = false;
    }
    //显示分页数量
    this.labBackPage.Text = Convert.ToString(ps.PageCount);
    //绑定 Repeater1 控件
```

```
this.Repeater1.DataSource = ps;
this.Repeater1.DataBind();
}
```

③ 按钮 InkbtnOne、InkbtnBack、InkbtnNext、InkbtnUp 的 Click 事件可参见 2.9.5 节内容。

2.9.10 用户注册界面UserReg.aspx

当在网站前台首页 Web/Default.aspx 页面中单击“新用户注册”时，将进入用户注册界面 UserReg.aspx 页面中。新用户可以在此页面进行用户注册，注册后就具有了发表新闻消息的权利，同时可以修改个人信息。

在“web”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“UserReg.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web”文件夹下的 UserReg.aspx 页面，进行页面设计。页面效果如图 2-16 所示。



图 2-16 UserReg.aspx 页面效果图

1. 前台页面设计

用户注册页面 UserReg.aspx，其页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 UserReg.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，其中最上面是自定义 Web 控件 Top.ascx，最下面是自定义 Web 控件 Bottom.ascx。在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-17 所示。

表 2-17 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label0	Text 属性设置为“新 用 户 注 册” Font 属性中的 Bold 设置为“True”	说明性描述
	Label1	Text 属性设置为“用户名称: ” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“注册用户名由大小写字母、 数字组成，长度限制为 3—12 字节”	说明性描述
	Label3	Text 属性设置为“用户密码: ” Font 属性中的 Bold 设置为“True”	说明性描述
	Label4	Text 属性设置为“请输入密码，区分大小写”	说明性描述
	Label5	Text 属性设置为“确认密码: ” Font 属性中的 Bold 设置为“True”	说明性描述
	Label6	Text 属性设置为“请再次输入密码”	说明性描述
	Label7	Text 属性设置为“电子邮箱: ” Font 属性中的 Bold 设置为“True”	说明性描述
	Label8	Text 属性设置为“请输入有效的邮件地址， 如 admin@.163.com.如果没有邮箱请先注册免 费邮箱”	说明性描述
标准/TextBox 控件	UserName	MaxLength 属性设置为“12”	要注册的用户名
	UserPwd1	TextMode 属性设置为“Password”	要使用的密码
	UserPwd2	TextMode 属性设置为“Password”	重新输入以确定 密码
	Email	使用默认设置	用户的邮箱
标准/Button 控件	CheckUser	Text 属性设置为“检测用户”	用于检测数据库 中是否已经存在了 申请的用户名
	Reg	Text 属性设置为“提 交” Enabled 属性设置为“False”	将申请的信息提 交到数据库中
验证 /RegularExpressionValida tor	RegularExpressionValidator1	ControlToValidate 属性设置为“UserName” ErrorMessage 属性设置为“*用户名格式不正 确!” ValidationExpression 属性设置为“^[a-zA-Z0 -9]+\$”	用户名格式的验证
验证 /RequiredFieldValidator	RequiredFieldValidator1	ControlToValidate 属性设置为“UserName” ErrorMessage 属性设置为“*用户名不能为空!”	用户名是否为空 验证
验证/CompareValidator	CompareValidator1	ControlToCompare 属性设置为“UserPwd1” ControlToValidate 属性设置为“UserPwd2” ErrorMessage 属性设置为“*两次输入密码不 一致!”	密码一致性验证
验证 /RegularExpressionValida tor	RegularExpressionValidator2	ControlToValidate 属性设置为“Email” ErrorMessage 属性设置为“*电子邮箱格式不正确!” ValidationExpression 属性设置为“\w+([+. \w+)*@\w+([-.]\w+)*.\w+([-.]\w+)*”	邮箱格式验证

2. 后台功能代码

在编辑器页（UserReg.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.UserInfo Ma = new Model.UserInfo();  
BLL.Userlogic Ba = new BLL.Userlogic();
```

程序主要代码如下。

① 按钮 Reg 的 Click 事件，执行后将提交用户注册的信息，其中用户权限为普通用户，并打开 UserCenter.aspx 页面。

```
protected void Reg_Click(object sender, EventArgs e)  
{  
    Ma.UserName= this.UserName.Text.Trim();  
    Ma.Password= FormsAuthentication.HashPasswordForStoringInConfigFile(this.  
UserPwd1.Text.Trim(), "MD5");  
    Ma.UserEmail = this.Email.Text.Trim();  
    Ma.Lever = "普通用户";  
    if (Ba.AddUser(Ma))  
    {  
        Response.Write("<script language=javascript>alert('注册成功！')</script>");  
    }  
    Session["username"] = Ma.UserName.ToString();  
    Response.Redirect("UserCenter.aspx");  
}
```

② 按钮 CheckUser 的 Click 事件，执行后将检测数据库中是否已经存在要注册的用户名。如果存在相同的用户名则不可以进行注册，否则可以进行正常的注册。

```
CheckUser_Click(object sender, EventArgs e)  
{  
    Ma.UserName = this.UserName.Text.Trim();  
    if (Ba.CheckUser(Ma) > 0)  
    {  
        Response.Write("<script language=javascript>alert('该用户已存在！')  
</script>");  
        this.UserName.Text = "";  
        this.Reg.Enabled = false;  
    }  
    else  
    {  
        Response.Write("<script language=javascript>alert('该用户可以注册！')  
</script>");  
        this.Reg.Enabled = true;  
    }  
}
```

2.9.11 用户发布新闻信息界面UserAddNews.aspx

当在网站前台首页 Web/Default.aspx 页面中以普通用户身份进行登录后，单击“发布新闻信息”将进入到发布新闻信息界面 UserAddNews.aspx 页面中。普通用户可以在此发布新闻信息，此信息需要经管理员审核后才能显示在网页上。

在“web”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“UserAddNews.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web”文件夹下的 UserAddNews.aspx 页面，进行页面设计。页面效果如图 2-17 所示。



图 2-17 UserAddNews.aspx 页面效果图

1. 前台页面设计

用户发布新闻信息页面 UserAddNews.aspx，其页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 UserAddNews.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，其中最上面是自定义 Web 控件 Top.ascx，最下面是自定义 Web 控件 Bottom.ascx。在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-18 所示。

表 2-18 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“发 布 新 闻” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“*新闻标题：”	说明性描述
	Label3	Text 属性设置为“*新闻类别：”	说明性描述
	Label4	Text 属性设置为“*新闻内容：”	说明性描述
	Label5	Text 属性设置为“*发布者：”	说明性描述
HTML/Input（Text）控 件	title	Size 属性设置为“30”	要发布的新闻标题
	user	Size 属性设置为“30” Value 属性设置为“admin”	新闻发布人的名字
标准/DropDownList 控 件	BigClassID	使用默认设置	显示新闻类别
验证 /RequiredFieldValidator 控件	title_check	ControlToVaidate 属性设置为“title” ErrorMessage 属性设置为“请输入标题”	验证新闻标题，新闻标题不 允许为空
	user_check	ControlToVaidate 属性设置为“user” ErrorMessage 属性设置为“请输入发布人”	验证新闻发布人，新闻发布 人不允许为空
自定义 FCKeditor 控件	FCKeditor1	BasePath 属性设置为“~/WenBenBianJiQi/fckeditor/”	用于编辑新闻内容
标准/Button 控件	Btn_OK	Text 属性设置为“提交”	将新闻信息提交到数据库， 等待管理员审核
	Reset	Text 属性设置为“重置”	清空已有的新闻信息

自定义控件 FCKeditor 是用于编辑新闻内容的文本编辑器，该控件采用现成的文本编辑器。将该文本编辑器添加为控件及其具体设置如下：

- ① 将该文本编辑器所在的“WenBenBianJiQi”文件夹复制到网站站点根目录下。
- ② 在该网站站点上，如“E:\News\News”上单击鼠标右键，选择“添加引用”，在“浏览”中找到“...\WenBenBianJiQi\Bin\FredCK.FCKeditorV2.dll”，单击“确定”按钮添加。
- ③ 在解决方案中双击网站站点下的 web.config 进行配置。添加代码如下：

```
<appSettings>  
  <add key="FredCK.FCKeditorV2:BasePath" value="/News/WenBenBianJiQi/fckeditor/"/>  
  <add key="FredCK.FCKeditorV2:UserFilePath" value="/News/WenBenBianJiQi/fckeditor/"/>  
  <add key="UploadFiles" value="/News/WenBenBianJiQi/fckeditor/"/>  
</appSettings>
```

- ④ 在工具箱的“常规”选项卡上单击鼠标右键，选择“选择项”。在选择工具箱中单击“浏览”按钮，找到“...\WenBenBianJiQi\Bin\FredCK.FCKeditorV2.dll”，然后单击“确定”按钮，完成添加自定义控件。
- ⑤ 将FCKeditor控件拖动到设计视图中，设置BasePath属性为“~/WenBenBianJiQi/fckeditor/”。
- ⑥ 此时需要建立文件夹，用以存储文本编辑器中上传的图片。在网站站点下的 Web 文件夹下建立 NewsImages 文件夹。

然后修改图片的上传路径。找到“WenBenBianJiQi”文件夹下“\fckeditor\editor\filemanager\connectors\aspx”中的 config.ascx 文件，双击将其打开，将该文件中的 UserFilesPath 设置为 UserFilesPath = "/Web/NewsImages/";。

- ⑦ 网站发布后或设置虚拟目录后浏览页面，即可实现文本编辑器功能。

2. 后台功能代码

在编辑器页（UserAddNews.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
BLL.BigClassLogic B_bc = new BLL.BigClassLogic();
Model.NewsInfo M_news = new Model.NewsInfo();
BLL.NewsLogic B_news = new BLL.NewsLogic();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Session["username"] == null)
        {
            Response.Write("<script language=javascript>alert('你还没有登录系统, 请返回首页登录系统! ');</script>");
        }
        else
        {
            DataBindDrownList();
            this.user.Value = Session["username"].ToString();
        }
    }
}
```

② 自定义方法 DataBindDrownList(), 将新闻分类绑定到控件上：

```
public void DataBindDrownList()
{
    DataSet ds = B_bc.GetBigClass();
    for (int i = 0; i < ds.Tables[0].DefaultView.Count; i++)
    {
        ListItem item = new ListItem();
        item.Text = ds.Tables[0].Rows[i]["name"].ToString();
        item.Value = ds.Tables[0].Rows[i]["B_id"].ToString();
        BigClassID.Items.Add(item);
        BigClassID.SelectedIndex = -1;
    }
}
```

③ 按钮 Btn_OK 的 Click 事件，执行后将新闻信息提交到数据库，等待管理员审核：

```
protected void Btn_OK_Click(object sender, EventArgs e)
{
    M_news.Title = this.title.Value.Trim();
    M_news.BigClassID = int.Parse(this.BigClassID.SelectedValue);
```

```

M_news.Info = this.FCKeditor1.Value.Trim();
M_news.UserName = this.user.Value.Trim();
if (B_news.AddNews(M_news))
{
    Response.Write("<script language='JavaScript'>if (confirm('按[确定]继续发布, 按[取消]回到系统首页'))");
    Response.Write("{ window.location = 'UserAddNews.aspx';}");
    Response.Write("else { window.location = 'Default.aspx';}</script>");
}
else
{
    Response.Write("<script language=javascript>alert('数据库操作有错误!');");
    Response.Write("</script>");
}
}

```

④ 按钮 Reset_Click 的 Click 事件, 执行后将清空已有的新闻信息:

```

protected void Reset_Click(object sender, EventArgs e)
{
    this.title.Value = "";
    this.FCKeditor1.Value = "";
}

```

2.9.12 个人管理信息页面UserCenter.aspx

当在网站前台首页 Web/Default.aspx 页面中以普通用户身份进行登录后, 单击“个人管理中心”将进入到个人管理信息界面 UserCenter.aspx 页面中。普通用户可以在此页面进行个人信息的修改, 并将结果提交到数据库中。

在“web”文件夹上右击, 选择“添加新项”, 然后选择“Web 窗体”并命名为“UserCenter.aspx”, 不要勾选“选择母版页”。单击“添加”按钮。添加后, 双击打开“web”文件夹下的 UserCenter.aspx 页面, 进行页面设计。页面效果如图 2-18 所示。

1. 前台页面设计

用户个人管理信息页面 UserCenter.aspx, 其页面设计的具体步骤如下:

首先, 将一个表格 (Table) 控件置于 UserCenter.aspx 页中, 为整个页面进行布局。然后, 从工具箱中拖动相应控件置于表格中, 其中最上面是自定义 Web 控件 Top.ascx, 最下面是自定义 Web 控件 Bottom.ascx。在各个控件上右击, 打开属性窗口, 设置控件的属性。“修改设置注册信息表”中各个控件的属性设置及用途如表 2-19 所示。



图 2-18 UserCenter.aspx 页面效果图

表 2-19 “修改设置注册信息表”中控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label0	Text 属性设置为“修 改 注 册 信 息” Font 属性中的 Bold 设置为“True”	说明性描述
	Label1	Text 属性设置为“用户名称: ” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“注册用户名由大小写字母、 数字组成, 长度限制为 3—12 字节”	说明性描述
	Label3	Text 属性设置为“用户密码: ” Font 属性中的 Bold 设置为“True”	说明性描述
	Label4	Text 属性设置为“请输入密码, 区分大小写”	说明性描述
	Label5	Text 属性设置为“确认密码: ” Font 属性中的 Bold 设置为“True”	说明性描述
	Label6	Text 属性设置为“请再次输入密码”	说明性描述
	Label7	Text 属性设置为“电子邮箱: ” Font 属性中的 Bold 设置为“True”	说明性描述
	Label8	Text 属性设置为“请输入有效的邮件地址,如 admin@.163.com.如果没有邮箱请先注册免费邮箱”	说明性描述

续表

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/TextBox 控件	TextBox1	MaxLength 属性设置为 “12”	要注册的用户名
	UserPwd1	TextMode 属性设置为 “Password”	要使用的密码
	UserPwd2	TextMode 属性设置为 “Password”	重新输入以确 定密码
	TextBox2	使用默认设置	用户的邮箱
标准/Button 控件	CheckUser	Text 属性设置为 “检测用户”	用于检测数据 库中是否已经存 在了申请的用户 名
	Reg	Text 属性设置为 “提 交” Enabled 属性设置为 “False”	将申请的信息 提交到数据库中
验证 /RegularExpressionValidator	RegularExpressionValidator1	ControlToValidate 属性设置为 “TextBox1” ErrorMessage 属性设置为 “*用户名格式不正确!” ValidationExpression 属性设置为 “^[a-zA-Z0-9] + \$”	用户名格式的 验证
验证 /RequiredFieldValidator	RequiredFieldValidator1	ControlToValidate 属性设置为 “TextBox1” ErrorMessage 属性设置为 “*用户名不能为空!”	用户名是否为空 验证
验证/CompareValidator	CompareValidator1	ControlToCompare 属性设置为 “UserPwd1” ControlToValidate 属性设置为 “UserPwd2” ErrorMessage 属性设置为 “*两次输入密码不一 致!”	密码一致性验证
验证 /RegularExpressionValidator	RegularExpressionValidator2	ControlToValidate 属性设置为 “TextBox2” ErrorMessage 属性设置为 “*电子邮箱格式不正 确!” ValidationExpression 属性设置为 “\w+([+.’]\w+) *@\w+([-.]\w+)*\.\w+([-.]\w+)*”	邮箱格式验证

“用户基本信息表” 中各个控件的属性设置及用途，如表 2-20 所示。

表 2-20 “用户基本信息表” 中控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/ Label 控件	Label11	Text 属性设置为 “今天是: ”	说明性描述
	Label12	Text 属性设置为 “用户名: ”	说明性描述
	Label13	Text 属性设置为 “用户邮箱: ”	说明性描述
	Label14	Text 属性设置为 “用户权限: ”	说明性描述
	TodayTime	Text 属性设置为 “ ” Width 属性设置为 “88px”	说明性描述
	UserName	Text 属性设置为 “ ”	显示用户名
	Email	Text 属性设置为 “ ” Width 属性设置为 “140px”	显示邮箱
	aleave	Text 属性设置为 “ ”	显示用户权限
标 准 / LinkButton 控件	LinkButton1	Text 属性设置为 “修改注册信息”	单击该按钮将显示用户要修 改的信息内容

2. 后台功能代码

在编辑器页（UserCenter.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
BLL.Userlogic Ba = new BLL.Userlogic();
Model.UserInfo Ma = new Model.UserInfo();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Session["username"] == null)
        {
            Response.Write("<script language=javascript>alert('数据库操作出错！')
</script>");
        }
        else
        {
            DataBindUserInfo();
        }
    }
}
```

② 自定义方法 DataBindUserInfo(), 实现数据的初始化：

```
public void DataBindUserInfo()
{
    Ma.UserName = Session["username"].ToString();
    DataSet ds = Ba.QueryUserInfoByName(Ma);
    this.Email.Text = ds.Tables[0].Rows[0][3].ToString();
    this.TextBox2.Text = ds.Tables[0].Rows[0][3].ToString();
    this.aLeave.Text = ds.Tables[0].Rows[0][4].ToString();
    this.UserName.Text = ds.Tables[0].Rows[0][1].ToString();
    this.TodayTime.Text = System.DateTime.Now.ToShortDateString().ToString();
}
```

③ 自定义方法 LinkButton1_Click (), 实现“数据信息表”内容的显示功能：

```
protected void LinkButton1_Click(object sender, EventArgs e)
{
    if (tb.Visible)
    {
        tb.Visible = false;
    }
    else
    {

```

```

        tb.Visible = true;
    }
}

```

④ 按钮 CheckUser 的 Click 事件, 执行后将检测数据库中是否已经存在要注册的用户名。如果存在相同的用户名则不可以进行注册, 否则可以进行正常的注册:

```

protected void CheckUser_Click(object sender, EventArgs e)
{
    Ma.UserName = this.TextBox1.Text.Trim();
    if (Ba.CheckUser(Ma) > 0)
    {
        Response.Write("<script language=javascript>alert('该用户已存在! ')");
        </script>");
        this.TextBox1.Text = "";
        this.Reg.Enabled = false;
    }
    else
    {
        Response.Write("<script language=javascript>alert('该用户可以注册! ')");
        </script>");
        this.Reg.Enabled = true;
    }
}

```

⑤ 按钮 Reg 的 Click 事件, 执行后将提交用户注册的信息, 其中用户权限为普通用户, 并打开 UserCenter.aspx 页面:

```

protected void Reg_Click(object sender, EventArgs e)
{
    Ma.UserName = this.TextBox1.Text.Trim();
    Ma.Password = FormsAuthentication.HashPasswordForStoringInConfigFile
(this.UserPwd1.Text.Trim(), "MD5");
    Ma.UserEmail = this.TextBox2.Text.Trim();
    Ma.Lever = "普通用户";
    if (Ba.AddUser(Ma))
    {
        Response.Write("<script language=javascript>alert('修改成功! ')</script>");
    }
    Session["username"] = Ma.UserName.ToString();
    Response.Redirect("UserCenter.aspx");
}

```

2.9.13 任务小结

本次任务用以实现网站的所有前台页面功能。通过完成本次任务, 要掌握页面的布局、各种控件的属性及应用, 熟练掌握数据控件与数据库之间的操作。

2.10 网站后台的实现

学习目标

- 进一步掌握数据控件的用法
- 掌握数据控件中添加服务器控件的方法及对其编写程序实现功能
- 建议学时：16 学时

2.10.1 任务名称：网站后台的实现

2.10.2 任务描述

网站后台的实现包括界面和功能的实现。

在后台管理中，管理员登录后可以对本系统进行后台的管理。管理员能够管理现有的新闻，可以对现有的新闻执行删除、修改、浏览等操作；同时，还可以发布新的新闻，对需要审核的新闻进行审核。要发布的新闻只有当管理员审核通过后才可以在新闻浏览页面，这样可以保证新闻的规范性。管理员还可以对新闻类别及系统用户进行增加、删除、修改等操作。由于一条新闻可以有多个评论，因此在新闻评论管理中，管理员可以删除一条新闻的单个评论，也可以删除一条新闻的全部评论。

网站后台的实现包括界面的实现和相关功能代码的实现。网站后台管理主要包括的功能有：

- 管理现有新闻；
- 发布新的新闻；
- 对要发布的新闻进行审核；
- 管理新闻评论；
- 管理新闻栏目；
- 管理系统用户。

2.10.3 任务分析

网站后台功能的设计也主要为数据访问层类、业务逻辑层类及表示层的各功能页，这三层都可以调用实体类。

在实现网站后台功能时要解决好各个页面的逻辑调用关系。

1. 实现表示层的页面对其他层功能的引用

本系统在数据层实现了对新闻、评论、用户、新闻类别的基本操作方法，在逻辑层中完成了对这些方法的逻辑调用，同时会根据各个页面的具体情况增加适当的方法。

在各个功能页面中都离不开对逻辑层相应类的对象的定义及使用，因此要根据不同的页面功能定义相应的对象，从而调用其方法实现功能。

2. 数据控件的应用

本系统后台中需要对数据进行大量操作，除了使用 Repeater 控件，还使用了 GridView 控件。

2.10.4 网站后台登录页面Admin_Login.aspx

在网站的“web”文件夹上右击，选择“新建文件夹”并命名为“adminManager”。在该

文件夹下将存放网站后台管理的各个功能页面。

在“web\adminManager”。文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_Login.aspx”。添加后，双击打开“web\adminManager”文件夹下的Admin_Login.aspx 页面，进行页面设计。其整体效果如图 2-19 所示。



图 2-19 登录页面效果图

网站后台登录页面的主要功能是验证用户权限，如果用户是管理员权限，则可以进入后台管理页面实现管理功能。

1. 前台页面设计

网站后台登录页面 Admin_Login.aspx 设计的具体步骤如下：

首先，将一个表格（Table）控件置于 Admin_Login.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-21 所示。

表 2-21 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	lbl_user	Text 属性设置为“用户：”	说明性描述
	lbl_password	Text 属性设置为“密码：”	说明性描述
标准/Textbox 控件	UserName	Height 属性设置为“14px” Width 属性设置为“130px”	输入管理员的用户名
	PassWord	Height 属性设置为“14px” Width 属性设置为“130px” TextMode 属性设置为“Password”	输入管理员的密码
标准/Button 控件	btn_Login	Text 属性设置为“登 录”	用户名密码验证通过后进入后台管理界面
	btn_Cancel	Text 属性设置为“重 置”	清空已输入的用户名、密码

2. 后台功能代码

在编辑器页（Admin_Login.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.UserInfo Ma = new Model.UserInfo();
BLL.Userlogic Ba = new BLL.Userlogic();
```

程序主要代码如下：

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
```



```

    {
}
    }

```

② 按钮 btn_Cancel_Click 的 Click 事件，执行后清空已输入的用户名、密码：

```

protected void btn_Cancel_Click(object sender, EventArgs e)
{
    UserName.Text = "";
    PassWord.Text = "";
    this.UserName.Focus();
}

```

③ 按钮 btn_Login_Click 的 Click 事件，用户名、密码验证通过后进入后台管理界面：

```

protected void btn_Login_Click(object sender, EventArgs e)
{
    if (UserName.Text.Equals(""))
    {
        Response.Write("<script language=javascript>alert('请输入管理员用户名！')");
        "</script>");
    }
    if (PassWord.Text.Equals(""))
    {
        Response.Write("<script language=javascript>alert('请输入管理员密码！')");
        "</script>");
    }
    Ma.UserName = UserName.Text.Trim();
    Ma.Password = FormsAuthentication.HashPasswordForStoringInConfigFile(
        PassWord.Text.Trim(), "MD5");
    if (Ba.AdminLogin(Ma) > 0)
    {
        Session["admin"] = UserName.Text.Trim();
        Response.Redirect("Admin_Index.aspx");
    }
    else
    {
        Response.Redirect("Admin_Login.aspx");
    }
}

```

2.10.5 网站后台管理首页Admin_Index.aspx

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_Index.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web\adminManager”文件夹下的 Admin_Index.aspx 页面，进行页面设计。

网站后台管理首页 Admin_Index.aspx 页面源视图中的完整代码如下：

```

<head runat="server">

```

```
<title>新闻后台管理</title>
</head>
<frameset rows="50,350,55" cols="*" frameborder="NO" border="0" framespacing="0">
  <frame src="Admin_top.aspx" name="top" scrolling="NO" noresize >
    <frameset cols="154,*" frameborder="NO" border="0" framespacing="0">
      <frame src="Admin_left.aspx" name="left" scrolling="NO" noresize>
        <frame src="Admin_right.aspx" name="main" scrolling=" YES " >
      </frameset>
    <frame src="Admin_bottom.aspx" name="bottom" scrolling=" NO " >
  </frameset>
</body>
  <form id="form1" runat="server">
    <div>
    </div>
  </form>
</body>
```

网站后台管理首页 Admin_Index.aspx，在该页面中需要用到 Admin_top.aspx、Admin_left.aspx、Admin_right.aspx、Admin_bottom.aspx4 个页面。

1. Admin_top.aspx页面

该页面效果如图 2-20 所示。



图 2-20 Admin_top.aspx 页面效果图

2. Admin_left.aspx页面

该页面效果如图 2-21 所示。



图 2-21 Admin_left.aspx 页面效果图

该页面需要在 Admin_left.aspx 页面源视图中编写代码。

(1) “新闻管理”菜单下各功能菜单代码

```

<tr>
    <td style="height:21px" onmouseover="style.backgroundColor='#d0dc0a'" onmouseout="style.backgroundColor
    =#42bd0b" bgcolor="#42bd0b"><div align="center"><a href="Admin_NewsList.aspx" target="
    main">管理现有新闻</a></div></td>
</tr>
<tr>
    <td onmouseover="style.backgroundColor='#d0dc0a'" onmouseout="style.backgroundColor='#42
    bd0b" bgcolor="#42bd0b" style="height: 21px"><div align="center"><a href="Admin_AddNews.aspx" target
    ="main">发布新闻内容</a></div></td>
</tr>
<tr>
    <td style="height:21px" onmouseover="style.backgroundColor='#d0dc0a'" onmouseout="style.backgroundColor
    =#42bd0b" bgcolor="#42bd0b"><div align="center"><a href="Admin_CheckNews.aspx" target="
    main">审核最新新闻</a></div></td>
</tr>
<tr>
    <td style="height:21px" onmouseover="style.backgroundColor='#d0dc0a'" onmouseout="style.backgroundColor
    =#42bd0b" bgcolor="#42bd0b"><div align="center"><a href="Admin_Comments.aspx" target="main">管理新闻
    评论</a></div></td>
</tr>

```

(2) “类别管理”菜单下的功能菜单代码

```

<tr>
    <td style="height:21px" onmouseover="style.backgroundColor='#d0dc0a'" onmouseout="style.backgroundColor
    =#42bd0b" bgcolor="#42bd0b"><div align="center"><a href="Admin_BigClass.aspx" target="main">管理新闻类
    别</a></div></td>
</tr>

```

(3) “用户管理”菜单下的功能菜单代码

```

<tr>
    <td style="height:21px" onmouseover="style.backgroundColor='#d0dc0a'" onmouseout="style.backgroundColor
    =#42bd0b" bgcolor="#42bd0b"><div align="center"><a href="Admin_AllUsers.aspx" target="main">管理系统用
    户</a></div></td>
</tr>

```

(4) “其他操作”菜单下的功能菜单代码

```

<tr>
    <td style="height:21px" onmouseover="style.backgroundColor='#d0dc0a'" onmouseout="
    style.backgroundColor
    =#42bd0b" bgcolor="#42bd0b"><div align="center"><a href="Admin_right.aspx" target="main">返回管理首
    页</a></div></td>
</tr>

```

```
<td style="height:21px" onmouseover="style.backgroundColor='#d0dc0a" onmouseout
="style.background
Color='#42bd0b" bgcolor="#42bd0b"><div align="center"><a href=" ../Default.aspx"target="_parent">返回系统首
页</a></div></td>
</tr>
```

3. Admin_right.aspx页面
该页面效果如图 2-22 所示。



图 2-22 Admin_right.aspx 页面效果图

4. Admin_bottom.aspx页面
该页面效果如图 2-23 所示。



图 2-23 Admin_bottom.aspx 页面效果图

2.10.6 网站后台管理现有新闻页面Admin_NewsList.aspx

在网站后台管理首页“web\adminManager\Admin_Index.aspx”页面中单击“管理现有新闻”，将在右侧框架中打开 管理现有新闻界面Admin_NewsList.aspx。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_NewsList.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web\adminManager”文件夹下的 Admin_NewsList.aspx 页面，进行页面设计。页面效果如图 2-24 所示。

1. 前台页面设计

网站后台管理现有新闻页面 Admin_NewsList.aspx，页面设计的具体步骤如下：



首先，将一个表格（Table）控件置于 Admin_NewsList.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-22 所示。

管理现有新闻

搜索新闻列表关键字：

按标题

搜索

序号	信息标题	所属分类	发布者	点击率	发布日期	审核状态		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	修改	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	修改	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	修改	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	修改	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	修改	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	修改	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	修改	删除

12

图 2-24 后台管理现有新闻页面效果图

表 2-22 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“管 理 现 有 新 闻” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“搜索新闻列表关键字：”	说明性描述
	BigClass	Text 属性设置为"Label"	通过后台代码，显示新闻栏目
标准/TextBox 控件	TextBox1	Height 属性设置为“14px” Width 属性设置为“120px”	输入要查询的关键字
标准/DropDownList 控件	DropDownList1	设置 Items 属性的成员为“按标题”和“按内容”	搜索的分类
数据/GridView 控件	GridView1	详细设置见下面	显示现有的所有新闻
标准/Button 控件	SearchNews	Text 属性设置为“搜索”	

在源视图中，在 GridView1 控件所在位置上方添加如下代码，其中 Label 控件用以显示新闻栏目：

```
<table style="width:640px; height:20px" width="640px" border="0" cellspacing="0" cellpadding="0" align="center">
    <tr style="width:100%">
        <asp:Label ID="BigClass" runat="server" Text="Label"></asp:Label>
    </tr>
</table>
```

GridView1 控件的设置如下：

```
<asp:GridView AllowPaging="True" AutoGenerateColumns="False" ID="GridView1"
    OnPageIndexChanging="GridView1_PageIndexChanging"
    OnRowDataBound="GridView1_RowDataBound" runat="server" Width="641px" PageSize="6">
    <Columns><asp:BoundField DataField="N_id" HeaderText="序号" />
    <asp:TemplateField HeaderText="信息标题">
        <ItemTemplate> <a href='../ListView.aspx?cid=<%#DataBinder.Eval(Container.
        DataItem,"N_id")%>' target="_blank" title="">
            <asp:Label ID="Label1" runat="server" Text='<%#CutString(DataBinder.Eval
            (Container.DataItem,"Title").ToString(),14)%>'></asp:Label></a>
        </ItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="所属分类">
        <ItemTemplate>
```

```

        <table align="center">
        <tr align="center">
        <td><a href='../BigTypeNews.aspx?sort=<%=#DataBinder.Eval(Container.DataItem,
"BigClassID")%>' target="_blank" title="">
<asp:Label ID="Label6" runat="server" Text='<%=# Bind("Name") %>'>
</asp:Label></a> </td> </tr></table>

        </ItemTemplate>
    </asp:TemplateField>
    <asp:BoundField DataField="UserName" HeaderText="发布者" />
    <asp:BoundField DataField="Hit" HeaderText="点击率" />
    <asp:BoundField DataField="InfoTime" HeaderText="发布日期" />
    <asp:BoundField DataField="Flag" HeaderText="审核状态" />
    <asp:TemplateField HeaderImageUrl="~/Web/images/edit.gif" HeaderText="修改">
        <ItemTemplate>
            <table align="center">
            <tr align="center">
            <td style="width: 25px; height: 14px">
<asp:Label ID="Label3" runat="server" ForeColor="Blue">
<a href="Admin_EditNews.aspx?cid=<%=# DataBinder.Eval
(Container.DataItem,"N_id") %>">修改</a></asp:Label></td>
</tr>

            </table>
        </ItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderImageUrl="~/Web/images/standard/delete.gif"
    HeaderText="删除">
        <ItemTemplate>
            <table align="center">
            <tr align="center">
            <td style="width: 25px; height: 14px">
                <asp:Label ID="Label3" runat="server" ForeColor="Blue">
<a href="Admin_DeleteNews.aspx?cid=<%=# DataBinder.Eval
(Container.DataItem,"N_id") %>">删除</a></asp:Label></td>
            </tr>
            </table>
        </ItemTemplate>
    </asp:TemplateField>
</Columns>
</asp:GridView>

```

2. 后台功能代码

在编辑器页（Admin_NewsList.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```

BLL.BigClassLogic B_bc = new BLL.BigClassLogic();
BLL.NewsLogic B_news = new BLL.NewsLogic();
DAL.FormatString fs = new DAL.FormatString();

```

程序主要代码如下。

① 在 Page_Load 事件中:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Session["admin"] == null)
        {
            Response.Redirect("Admin_Login.aspx");
        }
        DataSet sortDS = B_bc.GetBigClass();
        string content = "";
        for (int i = 0; i < sortDS.Tables[0].DefaultView.Count; i++)
        {
            content = content + "<td align='center' onmouseover=this.bgColor='FFFFFF'; onmouseout=this.bgColor='CCCCCC'; bgColor='#cccccc'>";
            content = content + "<a href='Admin_NewsList.aspx?sort=" + sortDS.Tables[0].Rows[i][0].ToString() + ">" + "<font size=2 color=#000000>" + sortDS.Tables[0].Rows[i][1].ToString() + "</font> " + "</a></td>";
        }
        content = content + "<td align='center' onmouseover=this.bgColor='FFFFFF'; onmouseout=this.bgColor='CCCCCC'; bgColor='#cccccc'>";
        content = content + "<a href='Admin_NewsList.aspx'>" + "<font size=2 color=#000000>重新排序" + "</font>" + "</a></td>";
        BigClass.Text = content;
        //分栏目获取新闻列表
        if (Request.QueryString["sort"] == null)
        {
            LoadNewInfo();
        }
        else
        {
            M_news.BigClassID = int.Parse(Request.QueryString["sort"].ToString());
            LoadNewInfo(M_news.BigClassID);
        }
    }
}
```

② QueryNews_Click 事件:

```
protected void QueryNews_Click(object sender, EventArgs e)
{
    //处理模糊查询方向
    if (this.DropDownList1.SelectedValue.ToString().Equals("title"))
    {
```

```

        M_news.Title = this.TextBox1.Text.Trim();
        this.GridView1.DataSource = B_news.AdminQueryByNewsTitle(M_news);
        this.GridView1.DataBind();
    }
    else
    {
        M_news.Info = this.TextBox1.Text.Trim();
        this.GridView1.DataSource = B_news.AdminQueryByNewsInfo(M_news);
        this.GridView1.DataBind();
    }
}

```

③ 自定义方法 LoadNewInfo(), 加载数据方法:

```

public void LoadNewInfo()
{
    GridView1.DataSource = B_news.GetData_news();
    GridView1.DataBind();
}

```

④ 自定义方法 LoadNewInfo(int BigClassID), 加载数据方法:

```

public void LoadNewInfo(int BigClassID)
{
    GridView1.DataSource = B_news.GetData_news(BigClassID);
    GridView1.DataBind();
}

```

⑤ 自定义带参方法 CutString(), 字符串截取方法:

```

public string CutString(string str, int len)
{
    return fs.CutString(str, len);
}

```

⑥ 方法 GridView1_RowDataBound, 进行数据颜色的绑定:

```

protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    //设置鼠标的指向行变效果
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        e.Row.Attributes.Add("onmouseover", "this.style.backgroundColor='#CCCCCC';this.style.color='#FFFFFF';this.style.cursor='#CCCCCC';");
        e.Row.Attributes.Add("onmouseout", "this.style.backgroundColor='#FFFFFF';this.style.color='#FFFFFF';");
    }
}

```


⑦ 方法 GridView1_PageIndexChanging, 进行新闻的分页绑定:

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    this.GridView1.PageIndex = e.NewPageIndex;
    LoadNewInfo();
}
```

2.10.7 网站后台修改新闻页面Admin_EditNews.aspx

网站后台 管理现有新闻页面 (Admin_NewsList.aspx) 中的 GridView1 控件, 其中的修改功能调用了Admin_EditNews.aspx页面。

在 “web\adminManager” 文件夹上右击, 选择 “添加新项”, 然后选择 “Web 窗体” 并命名为 “Admin_EditNews.aspx”, 不要勾选 “选择母版页”。单击 “添加” 按钮。添加后, 双击打开 “web\adminManager” 文件夹下的 Admin_EditNews.aspx 页面, 进行页面设计。页面效果如图 2-25 所示。



图 2-25 后台修改新闻页面效果图

1. 前台页面设计

网站后台修改新闻页面 Admin_EditNews.aspx, 页面设计的具体步骤如下:

首先, 将一个表格 (Table) 控件置于 Admin_EditNews.aspx 页中, 为整个页面进行布局。然后, 从工具箱中拖动相应控件置于表格中, 并在各个控件上右击, 打开属性窗口, 设置控件的属性。各个控件的属性设置及用途如表 2-23 所示。

表 2-23 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为 “修 改 新 闻” Font 属性中的 Bold 设置为 “True”	说明性描述
	Label2	Text 属性设置为 “*新闻标题: ”	说明性描述
	Label3	Text 属性设置为 “*新闻类别: ”	说明性描述
	Label4	Text 属性设置为 “*新闻内容: ”	说明性描述
	Label5	Text 属性设置为 “*发布人: ”	说明性描述

控 件 类 型	控 件 ID	主要属性设置	用 途
HTML/Input(Text)控件	title	Size 属性设置为“30”	要修改的新闻标题
	user	Size 属性设置为“30” Value 属性设置为“admin”	新闻发布人的名字
标准/DropDownList 控件	BigClassID	使用默认设置	显示新闻类别
验证 /RequiredFieldValidator 控 件	title_check	ControlToVaidate 属性设置为“title” ErrorMessage 属性设置为“请输入标题”	验证新闻标题，新闻标题不允许为空
	user_check	ControlToVaidate 属性设置为“user” ErrorMessage 属性设置为“请输入发布人”	验证新闻发布人，新闻发布人不允许为空
自定义 FCKeditor 控件	FCKeditor1	BasePath 属性设置为“~/WenBenBianJiQi/fckeditor/”	要修改的新闻内容
标准/Button 控件	Btn_OK	Text 属性设置为“提交”	将新闻信息提交到数据库，等待管理员审核
	Reset	Text 属性设置为“重置”	清空已有的新闻信息

2. 后台功能代码

在编辑器页（Admin_AddNews.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.NewsInfo M_news = new Model.NewsInfo();
BLL.BigClassLogic B_bc = new BLL.BigClassLogic();
BLL.NewsLogic B_news = new BLL.NewsLogic();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        DataBindDrownLst();
        Session["id"] = int.Parse(Request.QueryString["cid"].ToString());
        news.N_id = int.Parse(Request.QueryString["cid"].ToString());
        ataSet ds = B_news.DataBindNews(M_news.N_id);
        if (ds.Tables[0].Rows.Count > 0)
        {
            s.title.Value = ds.Tables[0].Rows[0][1].ToString();
            this.FCKeditor1.Value = ds.Tables[0].Rows[0][2].ToString();
            this.user.Value = ds.Tables[0].Rows[0][4].ToString();
        }
    }
}
```

② 自定义方法 DataBindDrownList(), 将新闻分类绑定到控件上：

```
public void DataBindDrownLst()
{
    DataSet ds = B_bc.GetBigClass();
    for (int i = 0; i < ds.Tables[0].DefaultView.Count; i++)
    {
```

```

        ListItem item = new ListItem();
        item.Text = ds.Tables[0].Rows[i]["name"].ToString();
        item.Value = ds.Tables[0].Rows[i]["B_id"].ToString();
        BigClassID.Items.Add(item);
        BigClassID.SelectedIndex = -1;
    }
}

```

③ 按钮 Btn_OK 的 Click 事件，执行后将新闻信息提交到数据库，等待管理员审核：

```

protected void Btn_OK_Click(object sender, EventArgs e)
{
    M_news.Title = this.title.Value.Trim();
    M_news.BigClassID = int.Parse(this.BigClassID.SelectedValue.ToString());
    M_news.Info = this.content.Value.Trim();
    M_news.UserName = this.user.Value.Trim();
    if (B_news.UpdateNews(M_news))
    {
        Response.Write("<script language='JavaScript'>if (confirm('按[确定]继续发布，按[取消]回到系统首页'))");
        Response.Write("{ window.location = 'Admin_NewsList.aspx';}");
        Response.Write("else { window.location = 'Default.aspx';}</script>");
    }
}

```

④ 按钮 Reset_Click 的 Click 事件，执行后将清空已有的新闻信息：

```

protected void Reset_Click(object sender, EventArgs e)
{
    this.title.Value = "";
    this.FCKeditor1.Value = "";
}

```

2.10.8 网站后台删除新闻页面Admin_DeleteNews.aspx

网站后台 管理现有新闻页面（Admin_NewsList）中的GridView1 控件，其中的删除功能调用了Admin_DeleteNews.aspx页面功能。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_DeleteNews”，不要勾选“选择母版页”。单击“添加”按钮。添加后，只需在编辑器页（Admin_DeleteNews.aspx.cs）编写代码。首先需要定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```

Model.NewsInfo M_news = new Model.NewsInfo();
BLL.NewsLogic B_news = new BLL.NewsLogic();
Model.CommentsInfo M_answer = new Model.CommentsInfo();
BLL.CommentsLogic B_answer = new BLL.CommentsLogic();
Model.BigClassInfo M_bigClass = new Model.BigClassInfo();
BLL.BigClassLogic B_bigClass = new BLL.BigClassLogic();

```

程序代码如下。

在 Page_Load 事件中，实现删除新闻功能：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Request.QueryString["cid"] == null)
        {
            Response.Write("<script language=javascript>alert('数据库操作出错,请正确操作!')</script>");
        }
        else
        {
            //执行删除操作,删除新闻及该新闻的全部评论信息
            M_news.N_id = int.Parse(Request.QueryString["cid"].ToString());
            M_answer.NewID = int.Parse(Request.QueryString["cid"].ToString());
            M_bigClass.B_id = int.Parse(B_news.GetBigClassIDByNewsID(M_news.N_id).
            Tables[0].Rows[0][0].ToString());
            if (B_news.DeleteNewsByID(M_news) && B_answer.DeleteAllByNewsID(M_answer))
            {
                B_bigClass.UpdateNewsCountDEL(M_bigClass);
                Response.Redirect("Admin_NewsList.aspx");
            }
        }
    }
}
```

2.10.9 网站后台发布新闻页面Admin_AddNews.aspx

在网站后台管理首页“web\adminManager\Admin_Index.aspx”页面中单击“添加新闻内容”，将在右侧框架中打开发布新闻信息界面Admin_AddNews.aspx。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_AddNews.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web\adminManager”文件夹下的 Admin_AddNews.aspx 页面，进行页面设计。页面效果如图 2-26 所示。

1. 前台页面设计

网站后台发布新闻页面 Admin_AddNews.aspx，页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 Admin_AddNews.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-24 所示。

发布新闻

新闻标题:

请输入标题

新闻类别:

未绑定

新闻内容:

FCKeditor V2 - FCKeditor1

发布人:

admin

请输入发布人

提交

重置

图 2-26 后台发布新闻页面效果图

表 2-24 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“发 布 新 闻” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“*新闻标题: ”	说明性描述
	Label3	Text 属性设置为“*新闻类别: ”	说明性描述
	Label4	Text 属性设置为“*新闻内容: ”	说明性描述
	Label5	Text 属性设置为“*发布人: ”	说明性描述
HTML/Input(Text)控件	title	Size 属性设置为“30”	要发布的新闻标题
	user	Size 属性设置为“30” Value 属性设置为“admin”	新闻发布人的名字
标准/DropDownList 控件	BigClassID	使用默认设置	显示新闻类别
验证/RequiredFieldValidator 控件	title_check	ControlToVaidate 属性设置为“title” ErrorMessage 属性设置为“请输入标题”	验证新闻标题, 新闻标题不允许为空
	user_check	ControlToVaidate 属性设置为“user” ErrorMessage 属性设置为“请输入发布人”	验证新闻发布人, 新闻发布人不允许为空
自定义FCKeditor 控件	FCKeditor1	BasePath 属性设置为“~/WenBenBianJiQi/fckeditor/”	要发布的新闻内容
标准/Button 控件	Btn_OK	Text 属性设置为“提交”	将新闻信息提交到数据库, 等待管理员审核
	Reset	Text 属性设置为“重置”	清空已有的新闻信息

2. 后台功能代码

在编辑器页（Admin_AddNews.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
BLL.BigClassLogic B_bc = new BLL.BigClassLogic();
Model.NewsInfo M_news = new Model.NewsInfo();
BLL.NewsLogic B_news = new BLL.NewsLogic();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
```

```

        {
            if (!Page.IsPostBack)
            {
                if (Session["username"] == null)
                {
                    Response.Write("<script language=javascript>alert('你还没有登录系统,请返回首页登录系统! ');</script>");
                }
                else
                {
                    DataBindDrownList();
                    this.user.Value = Session["username"].ToString();
                }
            }
        }
    }
}

```

② 自定义方法 DataBindDrownList(), 将新闻分类绑定到控件上:

```

public void DataBindDrownList()
{
    DataSet ds = B_bc.GetBigClass();
    for (int i = 0; i < ds.Tables[0].DefaultView.Count; i++)
    {
        ListItem item = new ListItem();
        item.Text = ds.Tables[0].Rows[i]["name"].ToString();
        item.Value = ds.Tables[0].Rows[i]["B_id"].ToString();
        BigClassID.Items.Add(item);
        BigClassID.SelectedIndex = -1;
    }
}

```

③ 按钮 Btn_OK 的 Click 事件, 执行后将新闻信息提交到数据库, 等待管理员审核:

```

protected void Btn_OK_Click(object sender, EventArgs e)
{
    M_news.Title = this.title.Value.Trim();
    M_news.BigClassID = int.Parse(this.BigClassID.SelectedValue.ToString());
    M_news.Info = this.FCKeditor1.Value.Trim();
    M_news.UserName = this.user.Value.Trim();
    if (B_news.AddNews(M_news))
    {
        Response.Write("<script language=JavaScript>if (confirm('按[确定]继续发布, 按[取消]回到系统首页'))");
        Response.Write("{window.location = 'UserAddNews.aspx'}");
        Response.Write("else {window.location = 'Default.aspx'}</script>");
    }
    else
    {

```

```
        Response.Write("<script language=javascript>alert('数据库操作有错误!');");
        Response.Write("</script>");
    }
}
```

④ 按钮 Reset_Click 的 Click 事件，执行后将清空已有的新闻信息：

```
protected void Reset_Click(object sender, EventArgs e)
{
    this.title.Value = "";
    this.FCKeditor1.Value = "";
}
```

2.10.10 网站后台审核新闻页面Admin_CheckNews.aspx

在网站后台管理首页“web\adminManager\Admin_Index.aspx”页面中单击“审核最新新闻”，将在右侧框架中打开审核新闻界面Admin_CheckNews.aspx。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_CheckNews.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web\adminManager”文件夹下的 Admin_CheckNews.aspx 页面，进行页面设计。页面效果如图 2-27 所示。

审 核 最 新 新 闻						
编号	新闻标题	所属栏目	发布者	发布时间	新闻状态	
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	进行审核
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	进行审核
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	进行审核
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	进行审核
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	进行审核

没有需要审核的新闻!

图 2-27 后台审核新闻页面效果图

1. 前台页面设计

网站后台审核新闻页面 Admin_CheckNews.aspx，页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 Admin_CheckNews.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-25 所示。

表 2-25 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“审 核 最 新 新 闻” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“没有需要审核的新闻！” Font 属性中的 Size 设置为“Medium”	说明性描述
数据/GridView 控件	GridView1	详细设置见下面	显示需要审核的新闻信息，提供审核功能

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" Width="540px"
OnRowDataBound="GridView1_RowDataBound" Height="184px">
<Columns>
<asp:TemplateField HeaderText="编号">
<ItemTemplate>
<table align="center">
<tr align="center">
<td>
<asp:Label ID="Label2" runat="server" Text='<%= Bind("N_id") %>'>
</asp:Label></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="新闻标题">
<ItemTemplate>
<table align="center">
<tr align="center">
<td><a href='../ListView.aspx?cid=<%= #DataBinder.Eval(Container.DataItem, "N_id") %>' title=""
target="_blank"><asp:Label ID="Label3" runat="server" Text='<%= Bind("Title") %>'></asp:Label>
</a></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="所属栏目">
<ItemTemplate>
<table align="center">
<tr align="center">
<td><a href='../BigTypeNews.aspx?sort=<%= #DataBinder.Eval(Container.
DataItem, "BigClassID") %>' title="" target="_blank"><asp:Label ID=
"Label7" runat="server" Text='<%= Bind("Name") %>'></asp:Label></a>
</td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="发布者">
<ItemTemplate>
<table align="center">
<tr align="center">
<td><asp:Label ID="Label4" runat="server" Text='<%= Bind("UserName") %>'>
</asp:Label></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
```



```

</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="发布时间">
<ItemTemplate>
<table align="center">
<tr align="center">
<td><asp:Label ID="Label5" runat="server" Text='<%= Convert.ToDateTime
(DataBinder.Eval(Container.DataItem,"InfoTime")).ToShortDateString() %
>'></asp:Label></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="新闻状态">
<ItemTemplate>
<table align="center">
<tr align="center">
<td><asp:Label ID="Label1" runat="server" Text='<%= Bind("Flag") %>'>
</asp:Label></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderImageUrl="~/Web/admin/Editor/ButtonImage/standard/editmenu.gif"
HeaderText="操作">
<ItemTemplate>
<table align="center">
<tr align="center">
<td><a href='./adminManager/CheckNews.aspx?cid=<%=DataBinder.Eval
(Container.DataItem,"N_id")%>&sort=<%=DataBinder.Eval(Container.
DataItem,"BigClassID")%>'><asp:Label ID="Label6" runat="server">
进行审核</asp:Label></a>
</td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
</Columns>
</asp:GridView>

```

2. 后台功能代码

在编辑器页（Admin_CheckNews.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
BLL.NewsLogic B_news = new BLL.NewsLogic();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Session["admin"] == null)
        {
            Response.Redirect("Admin_Login.aspx");
        }
        if (GetDS().Tables[0].Rows.Count > 0)
        {
            this.Label2.Visible = false;
        }
        else
        {
            his.Label2.Visible = true;
        }
        LoadData();
    }
}
```

② 自定义方法 LoadData(), 用以初始化数据:

```
public void LoadData()
{
    this.GridView1.DataSource = B_news.GetNewsOfFlag0();
    this.GridView1.DataBind();
}
```

③ 自定义方法 GetDS(), 获取没有通过审核的数据集合:

```
public DataSet GetDS()
{
    return B_news.GetNewsOfFlag0();
}
```

④ 方法 GridView1_RowDataBound(), 用以控制选中数据行的颜色变化:

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        e.Row.Attributes.Add("onmouseover", "this.style.backgroundColor='#CCCCCC';this.style.color='#FFFFFF';this.style.cursor='#CCCCCC';");
        e.Row.Attributes.Add("onmouseout", "this.style.backgroundColor='#FFFFFF';this.style.color='#FFFFFF';");
    }
}
```

2.10.11 网站后台 管理审核功能页面CheckNews.aspx

网站后台管理审核新闻（Admin_CheckNews.aspx）中的 GridView1 控件，其中的审核功能调用了 CheckNews.aspx 的页面功能。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“CheckNews.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，只需在编辑器页（CheckNews.aspx.cs）编写代码。首先需要定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.NewsInfo M_news = new Model.NewsInfo();
BLL.NewsLogic B_news = new BLL.NewsLogic();
Model.BigClassInfo M_bc = new Model.BigClassInfo();
BLL.BigClassLogic B_bc = new BLL.BigClassLogic();
```

程序代码如下。

在 Page_Load 事件中，实现删除用户功能：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Request.QueryString["cid"] == null)
        {
            Response.Write("<script language=javascript>alert('数据库操作出错,请正确操作!')</script>");
        }
        else
        {
            M_news.N_id = int.Parse(Request.QueryString["cid"].ToString());
            M_bc.B_id = int.Parse(Request.QueryString["sort"].ToString());
            //新闻通过审核的同时设置该新闻有效，即正式为系统某栏目下的新闻，且该栏目下的新闻总数+ 1
            if (B_news.CheckNewsByID(M_news) && B_bc.UpdateNewsCount(M_bc))
            {
                Response.Write("<script language=javascript>alert('审核成功,单击[确定]返回审核页面!')</script>");
                Response.Redirect("Admin_CheckNews.aspx");
            }
            else
            {
                Response.Write("<script language=javascript>alert('审核失败!')</script>");
            }
        }
    }
}
```

2.10.12 网站后台管理新闻评论页面Admin_Comments.aspx

在网站后台管理首页“web\adminManager\Admin_Index.aspx”页面中单击“管理新闻评论”，将在右侧框架中打开管理新闻评论界面 Admin_Comments.aspx。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_Comments.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web\adminManager”文件夹下的 Admin_Comments.aspx 页面，进行页面设计。页面效果如图 2-28 所示。

新闻评论管理

查询新闻评论 (请选择新闻ID) :

请选择

查询

查询全部

选择	评论编号	评论用户	评论内容	评论时间	新闻编号
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
<input type="checkbox"/>	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
1 2					
全部选择	删除选中项	删除选定新闻ID的新闻的全部评论			

图 2-28 后台管理新闻评论页面效果图

1. 前台页面设计

网站后台管理新闻评论页面 Admin_Comments.aspx，页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 Admin_Comments.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-26 所示。

表 2-26 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“新 闻 评 论 管 理” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“查询新闻评论(请选择新闻 ID): ”	说明性描述
标准/DropDownList 控件	DlnewsID	设置 Items 属性的成员为“请选择”	显示新闻 ID
标准/GridView	GridView1	详细设置见下面	显示新闻评论
标准/Button 控件	SearchByNewID	Text 属性设置为“查询”	通过选定的新闻 ID 查询该条新闻对应的评论
	SeachAll	Text 属性设置为“查询全部”	查询所有的新闻评论
	SelectAll	Text 属性设置为“全部选择”	选择所有的新闻评论
	DeleteCheck	Text 属性设置为“删除选中项”	删除选定的新闻评论
	DeleteAll	Text 属性设置为“删除选定新闻 ID 的新闻的全部评论”	删除选定新闻 ID 的新闻的全部评论

控件 GridView1 的设置如下：

```
<asp:GridView ID="GridView1" runat="server" Width="642px" AutoGenerateColumns="False"
Height="112px" AllowPaging="True" OnPageIndexChanging="GridView1_PageIndexChanging"
OnRowDataBound="GridView1_RowDataBound">
<Columns>
    <asp:TemplateField HeaderText="选择">
        <ItemTemplate>
            <table align="center">
                <tr align="center">
                    <td>
                        <asp:CheckBox ID="CheckBox1" runat="server" /></td> </tr>
            </table>
        </ItemTemplate>
    </asp:TemplateField>
    <asp:BoundField DataField="C_id" HeaderText="评论编号" />
    <asp:BoundField DataField="C_user" HeaderText="评论用户" />
    <asp:BoundField DataField="C_word" HeaderText="评论内容" />
    <asp:BoundField DataField="C_time" HeaderText="评论时间" />
    <asp:BoundField DataField="NewsID" HeaderText="新闻编号" />
</Columns><PagerStyle ForeColor="Black" />
</asp:GridView>
```

2. 后台功能代码

在编辑器页（Admin_Comments.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.CommentsInfo Ma = new Model.CommentsInfo();
DAL.FormatString fs = new DAL.FormatString();
BLL.CommentsLogic Ba = new BLL.CommentsLogic();
BLL.NewsLogic B_news = new BLL.NewsLogic();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Session["admin"] == null)
        {
            Response.Redirect("Admin_Login.aspx");
        }
        DataBindAnswer();
        DataBindDList();
        DeleteAll.Attributes.Add("onclick", "return confirm('真的要删除全部记录吗? ');");
    }
}
```

② 自定义方法 `DataBindDLList()`，用以显示新闻 ID：

```
public void DataBindDLList()
{
    this.DLnewsID.DataSource = B_news.GetNewsID();
    this.DLnewsID.DataTextField = "N_id";
    this.DLnewsID.DataValueField = "N_id";
    this.DLnewsID.DataBind();
}
```

③ 自定义方法 `DataBindComments()`，用以显示所有的评论：

```
public void DataBindComments()
{
    this.GridView1.DataSource = Ba.GetAllAnswer();
    this.GridView1.DataBind();
}
```

④ 自定义方法 `CutString(string str, int len)`，用以截取字符串：

```
public string CutString(string str, int len)
{
    return fs.CutString(str, len);
}
```

⑤ 按钮 `SelectAll` 的 `Click` 事件，用以选取所有的评论：

```
protected void SelectAll_Click(object sender, EventArgs e)
{
    foreach (GridViewRow gvr in GridView1.Rows)
    {
        CheckBox cb = (CheckBox)gvr.Cells[0].FindControl("CheckBox1");
        if (cb.Checked)
        { }
    }
    else
    {
        cb.Checked = true;
    }
}
```

⑥ 按钮 `SearchByNewID` 的 `Click` 事件，用以选取选定新闻 ID 新闻的所有评论：

```
protected void SearchByNewID_Click(object sender, EventArgs e)
{
    Ma.NewID = int.Parse(this.DLnewsID.SelectedValue.ToString());
    this.GridView1.DataSource = Ba.GetAllCommentsByNewsID(Ma.NewID);
    this.GridView1.DataBind();
}
```

⑦ 按钮 SearchAll 的 Click 事件，用以选取所有新闻的所有评论：

```
protected void SearchAll_Click(object sender, EventArgs e)
{
    DataBindComments();
    Response.Redirect("Admin_Comments.aspx");
}
```

⑧ GridView1_RowDataBound()事件，用以控制选中数据行的颜色变化：

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    //设置鼠标的指向行变效果
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        e.Row.Attributes.Add("onmouseover", "this.style.backgroundColor='#CCCCCC';this.style.color='#FFFFFF';this.style.cursor='#CCCCCC';");
        e.Row.Attributes.Add("onmouseout", "this.style.backgroundColor='#FFFFFF';this.style.color='#FFFFFF';");
    }
}
```

⑨ GridView1_PageIndexChanging 事件，用以绑定分页：

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    this.GridView1.PageIndex = e.NewPageIndex;
}
```

⑩ 按钮 DeleteAll 的 Click 事件，删除选定新闻 ID 的新闻的所有评论：

```
protected void DeleteAll_Click(object sender, EventArgs e)
{
    Ma.NewID = int.Parse(this.DLnewsID.SelectedValue.ToString());
    Ba.DeleteAllByNewsID(Ma);
    DataBindComments();
    Response.Redirect("Admin_Comments.aspx");
}
```

⑪ 按钮 DeleteCheck 的 Click 事件，删除复选框选中的新闻评论：

```
protected void DeleteCheck_Click(object sender, EventArgs e)
{
    bool F = false;
    for(int i=0;i<GridView1.Rows.Count;i++)
    {
        CheckBox cb = (CheckBox)GridView1.Rows[i].Cells[0].FindControl("CheckBox1");
        if (cb.Checked==true)
        {
            Ma.C_id = int.Parse(GridView1.Rows[i].Cells[1].Text.ToString());
            F = true;
        }
    }
}
```

```
Ba.DeleteCommentsByCommentsID(Ma);
    }
    }
    this.DataBindComments();
    if (!F)
    {
        Response.Write("<script language=javascript>alert('你没有选择任何项！')
</script>");
    }
}
```

2.10.13 网站后台 管理新闻类别页面Admin_BigClass.aspx

在网站后台管理首页“web\adminManager\Admin_Index.aspx”页面中单击“管理新闻类别”，将在右侧框架中打开 管理新闻类别界面Admin_BigClass.aspx。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_BigClass.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web\adminManager”文件夹下的 Admin_BigClass.aspx 页面，进行页面设计。页面效果如图 2-29 所示。



图 2-29 后台 管理新闻类别页面效果图

1. 前台页面设计

网站后台 管理新闻类别页面Admin_BigClass.aspx，页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 Admin_BigClass.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-27 所示。

表 2-27 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“管 理 新 闻 类 别” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“添 加 新 闻 类 别”	说明性描述
	Label3	Text 属性设置为“类别名称： ”	说明性描述
数据/GridView 控件	GridView1	详细设置见下面	显示现有新闻类别

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/LinkButton 控件	LinkButton1	Text 属性设置为 “添加新的新闻类别”	显示添加新闻类别的页面
标准/TextBox 控件	BigClassName	Width 属性设置为 “126px”	输入新闻类别
验证/RequiredFieldValidator 控件	RequiredFieldValidator1	ControlToValidate 属性设置为 “BigClassName” ErrorMessage 属性设置为 “*请输入类别名称！”	不允许输入的新闻类别为空
标准/Button 控件	AddBigClassName	Text 属性设置为 “确定添加”	添加新闻类别到数据库

控件 GridView1 的具体设置如下：

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" Width="550px"
OnRowDataBound="GridView1_RowDataBound" Height="140px">
<Columns>
<asp:TemplateField HeaderText="栏目编号">
<ItemTemplate>
<table align="center">
<tr align="center">
<td>
<asp:Label ID="Label1" runat="server" Text='<%# Bind("B_id") %>'>
</asp:Label></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="栏目名称">
<ItemTemplate>
<table align="center">
<tr align="center">
<td><asp:Label ID="Label2" runat="server" Text='<%# Bind("Name") %>'>
</asp:Label></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="栏目属性">
<ItemTemplate>
<table align="center">
<tr align="center">
<td><asp:Label ID="Label3" runat="server" Text='<%# Bind("Flag") %>'>
</asp:Label></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="修改栏目">
<ItemTemplate>
<table align="center">
```

```

        <tr align="center">
        <td>
        <asp:Label ID="Label5" runat="server">
        <a href="Admin_EditBig.aspx?cid=<%=# DataBinder.Eval(Container.DataItem,
        "B_id") %>">修改</a></asp:Label></td>
        </tr>
    </table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="删除栏目">
    <ItemTemplate>
    <table align="center">
    <tr align="center">
    <td>
    <asp:Label ID="Label4" runat="server"><a href="Admin_DeleteBig.aspx? cid=
    <%=# DataBinder.Eval(Container.DataItem,"B_id") %>">删除</a></asp:Label></td>
    </tr>
    </table>
    </ItemTemplate>
</asp:TemplateField>
</Columns></asp:GridView>

```

2. 后台功能代码

在编辑器页（Admin_BigClass.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```

Model.BigClassInfo Mb = new Model.BigClassInfo();
BLL.BigClassLogic Bb = new BLL.BigClassLogic();

```

程序主要代码如下。

① 在 Page_Load 事件中：

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Session["admin"] == null)
        {
            Response.Redirect("Admin_Login.aspx");
        }
        LoadData();
    }
}

```

② 自定义方法 LoadData()，用以显示现有新闻类别：

```

public void LoadData()
{
    this.GridView1.DataSource = Bb.GetData_BigClass();
}

```

```

        this.GridView1.DataBind();
    }

```

③ GridView1 控件的 RowDataBound 事件，用以改变数据行的颜色：

```

protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        e.Row.Attributes.Add("onmouseover", "this.style.backgroundColor='#CCCCCC';this.style.color='#FFFFFF';this.style.cursor='#CCCCCC';");
        e.Row.Attributes.Add("onmouseout", "this.style.backgroundColor='#FFFFFF';this.style.color='#FFFFFF';");
    }
}

```

④ AddBigClassName 控件的 Click 事件，用以添加类别到数据库：

```

protected void AddBigClassName_Click(object sender, EventArgs e)
{
    Mb.Name = this.BigClassName.Text.Trim();
    if (Bb.AddBigClass(Mb))
    {
        Response.Write("<script language=javascript>alert('添加成功！')</script>");
    }
    this.BigClassName.Text = "";
    this.BigClassName.Focus();
    LoadData();
}

```

⑤ LinkButton1 控件的 Click 事件，用以显示要添加类别的界面：

```

protected void LinkButton1_Click(object sender, EventArgs e)
{
    if (this.tb_AddBigClassName.Visible == true)
    {
        this.tb_AddBigClassName.Visible = false;
    }
    else
    {
        this.tb_AddBigClassName.Visible = true;
    }
}

```

2.10.14 网站后台 管理修改新闻类别页面Admin_EditBig.aspx

网站后台 管理新闻类别页面（Admin_BigClass.aspx）中的 GridView1 控件，其中的修改功能调用了Admin_EditBig.aspx页面。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并

命名为“Admin_EditBig.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web\adminManager”文件夹下的 Admin_EditBig.aspx 页面，进行页面设计。页面效果如图 2-30 所示。

1. 前台页面设计

网站后台 管理修改新闻类别页面Admin_EditBig.aspx，页面设计的具体步骤如下：



图 2-30 后台 管理修改新闻类别页面效果图

首先，将一个表格（Table）控件置于 Admin_EditBig.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-28 所示。

表 2-28 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“修 改 新 闻 类 别” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“新闻编号：”	说明性描述
	Label3	Text 属性设置为“类别名称：”	说明性描述
	Label4	Text 属性设置为“是否可见：”	说明性描述
	Label5	Text 属性设置为“back”	说明性描述
	BigClassid	Text 属性设置为“ ”	显示新闻类别的 ID 编号
标准/TextBox 控件	BigClassName	使用默认设置	输入新闻类别名称
标 准 /DropDownList 控件	IsVisble	设置 Items 属性的成员分别为“显示”和“隐藏”	用于设置该新闻类别是否可见
标准/Button 控件	Edit	Text 属性设置为“确定修改”	把修改信息添加到数据库
标准/ImageButton 控件	ImageButton1	ImageUrl 设置为“~/Web/images/gif026.gif” PostBackUrl 设置为“~/Web/adminManager/Admin_BigClass.aspx”	返 回 Admin_BigClass.aspx 页面

2. 后台功能代码

在编辑器页（Admin_EditBig.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.BigClassInfo Mb = new Model.BigClassInfo();
BLL.BigClassLogic Bb = new BLL.BigClassLogic();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Request.QueryString["cid"] == null)
        {
            Response.Write("<script language=javascript>alert  
(数据库操作出错,请正确操作! ')</script>");
        }
        else
        {
            DataBindText(int.Parse(Request.QueryString["cid"].ToString()));
        }
    }
}
```

② 自定义方法 DataBindText(), 用以获取要修改的新闻类别及类别编号：

```
public void DataBindText(int id)
{
    DataSet ds = Bb.GetBigClassByID(id);
    this.BigClassid.Text = ds.Tables[0].Rows[0][0].ToString();
    this.BigClassName.Text = ds.Tables[0].Rows[0][1].ToString();
}
```

③ Edit 控件的 Click 事件，用于把修改信息添加到数据库中：

```
protected void Edit_Click(object sender, EventArgs e)
{
    Mb.B_id = int.Parse(this.BigClassid.Text.Trim());
    Mb.Name = this.BigClassName.Text.Trim();
    Mb.Flag = this.isVisble.SelectedValue.Trim();
    if (Bb.UpdateBigClassNameAndFlag(Mb))
    {
        Response.Redirect("Admin_BigClass.aspx");
    }
}
```

2.10.15 网站后台 管理删除新闻类别页面Admin_DeleteBig.aspx

网站后台 管理新闻类别页面（Admin_BigClass.aspx）中的 GridView1 控件，其中的删除功能调用了Admin_DeleteBig.aspx的页面功能。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_DeleteBig.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，只需在编辑器页（Admin_DeleteBig.aspx.cs）中编写代码。首先需要定义相关类的对象，以便在编

写代码时调用该类中的方法。代码如下：

```
Model.BigClassInfo Mb = new Model.BigClassInfo();
BLL.BigClassLogic Bb = new BLL.BigClassLogic();
Model.NewsInfo M_news = new Model.NewsInfo();
BLL.NewsLogic B_news = new BLL.NewsLogic();
Model.CommentsInfo M_answer = new Model.CommentsInfo();
BLL.CommentsLogic B_answer = new BLL.CommentsLogic();
```

程序主要代码如下。

在 Page_Load 事件中，实现删除新闻类别功能：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Request.QueryString["cid"] == null)
        {
            Response.Write("<script language=javascript>alert('数据库操作出错,请正确操作!')</script>");
        }
        else
        {
            Mb.B_id = int.Parse(Request.QueryString["cid"].ToString());
            M_news.BigClassID = int.Parse(Request.QueryString["cid"].ToString());
            M_answer.NewID = int.Parse(Request.QueryString["cid"].ToString());
            //删除新闻栏目
            if (Bb.DeleteBigClassByID(Mb))
            {
                //删除新闻栏目的同时删除该栏目对应的新闻信息及该新闻的全部评论信息
                if (B_news.DeleteNewsByBigClassID(M_news)&&B_answer.
DeleteAllByNewsID(M_answer))
                {
                    Response.Redirect("Admin_BigClass.aspx");
                }
            }
            else
            {
                Response.Write("<script language=javascript>alert('数据库操作出错,请正确操作!')</script>");
            }
        }
        else
        {
            Response.Write("<script language=javascript>alert('数据库操作出错,请正确操作!')</script>");
        }
    }
}
```

2.10.16 网站后台 管理系统用户页面Admin_AllUsers.aspx

在网站后台管理首页“web\adminManager\Admin_Index.aspx”页面中单击“管理系统用户”，将在右侧框架中打开 管理系统用户界面Admin_AllUsers.aspx。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_AllUsers.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web\adminManager”文件夹下的 Admin_AllUsers.aspx 页面，进行页面设计。页面效果如图 2-31 所示。



图 2-31 后台 管理系统用户页面效果图

1. 前台页面设计

网站后台 管理系统用户页面Admin_AllUsers.aspx，页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 Admin_AllUsers.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-29 所示。

表 2-29 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“管 理 系 统 用 户” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“添加用户信息” Font 属性中的 Bold 设置为“True”	说明性描述
	Label3	Text 属性设置为“用户名称： ”	说明性描述
	Label4	Text 属性设置为“用户密码： ”	说明性描述
	Label5	Text 属性设置为“用户邮箱： ”	说明性描述
	Label6	Text 属性设置为“用户权限： ”	说明性描述
数据/GridView 控件	GridView1	详细设置见下面	显示系统用户信息

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/LinkButton 控件	AddUserSet	Text 属性设置为“添加新用户信息”	显示添加新用户信息的页面
标准/TextBox 控件	UserName	Width 属性设置为“120px”	输入用户名称
	UserPwd	Width 属性设置为“120px”	输入用户密码
	UserEmail	Width 属性设置为“120px”	输入用户邮箱
验证 /RequiredFieldValidator 控 件	RequiredFieldValidator1	ControlToValidate 属性设置为“UserName” ErrorMessage 属性设置为“*用户名不能为空！”	不允许输入的用户名称为空
	RequiredFieldValidator2	ControlToValidate 属性设置为“UserPwd” ErrorMessage 属性设置为“*用户密码不能为空！”	不允许输入的用户密码为空
	RequiredFieldValidator3	ControlToValidate 属性设置为“UserEmail” ErrorMessage 属性设置为“*用户邮箱不能为空！”	不允许输入的用户邮箱为空
验证/ RegularExpressionValidator 控件	RegularExpressionValidator1	ControlToValidate 属性设置为“UserEmail” ErrorMessage 属性设置为“*电子邮箱格式不正确！” ValidationExpression 属性设置为“\w+([+.\] \\w+)*@[w+([+.\] \\w+)*\. \\w+([+.\] \\w+)*”	验证邮箱格式是否正确
标准/DropDownList 控件	DownListAleave	设置 Items 属性的成员分别为“管理员”和“普通用户”	用于选取系统管理权限
标准/Button 控件	btnAdd	Text 属性设置为“添 加”	将用户信息添加到数据库

GridView1 控件的详细设置如下：

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" Width="550px"
OnRowDataBound="GridView1_RowDataBound" Height="140px">
  <Columns>
    <asp:TemplateField HeaderText="用户编号">
      <EditItemTemplate>
        <asp:TextBox ID="TextBox1" runat="server" Text='<%# Bind("U_id") %>'></asp:TextBox>
      </EditItemTemplate>
      <ItemTemplate>
        <table align="center">
          <tr align="center">
            <td>
              <asp:Label ID="Label3" runat="server" Text='<%# Bind("U_id") %>'></asp:Label></td>
            </tr>
          </table>
        </ItemTemplate>
      </asp:TemplateField>
      <asp:TemplateField HeaderText="用户帐号">
        <EditItemTemplate>
          <asp:TextBox ID="TextBox2" runat="server" Text='<%# Bind("UserName") %>'></asp:TextBox>
        </EditItemTemplate>
        <ItemTemplate>
          <table align="center">
            <tr align="center">
```



```

<td>
<asp:Label ID="Label4" runat="server" Text='<%# Bind("UserName") %>'>
</asp:Label></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="电子邮箱">
<ItemTemplate><table align="center">
<tr align="center">
<td >
<asp:Label ID="Label5" runat="server" Text='<%# Bind("Email") %>'></asp:Label>
</td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="用户权限">
<ItemTemplate>
<table align="center">
<tr align="center">
<td>
<asp:Label ID="Label6" runat="server" Text='<%# Bind("Lever") %>'></asp:Label>
</td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="修 改">
<ItemTemplate>
<table align="center">
<tr align="center">
<td>
<asp:Label ID="Label2" runat="server" Width="25px"><a
href="Admin_EditUser.aspx?cid=<%#DataBinder.Eval(Container.DataItem,"U_id")%>">修改
</a></asp:Label></td>
</tr>
</table>
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="删 除">
<ItemTemplate>
<table align="center">
<tr align="center">
<td>
<asp:Label ID="Label1" runat="server"><a href="Admin_DeleteUser.aspx?cid=
<%#DataBinder.Eval(Container.DataItem,"U_id")%>">删除</a></asp:Label></td>

```

```

        </tr>
    </table>
</ItemTemplate>
</asp:TemplateField>
</Columns>
</asp:GridView>

```

2. 后台功能代码

在编辑器页（Admin_AllUsers.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```

Model.UserInfo Ma = new Model.UserInfo();
BLL.Userlogic ba = new BLL.Userlogic();

```

程序主要代码如下。

① 在 Page_Load 事件中：

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Session["admin"] == null)
        {
            Response.Redirect("Admin_Login.aspx");
        }
        LoadData();
    }
}

```

② 自定义方法 LoadData()，用以显示所有的用户：

```

public void LoadData()
{
    this.GridView1.DataSource = ba.GetDataAdmin();
    this.GridView1.DataBind();
}

```

③ AddUserSet 控件的 Click 事件，用以显示添加用户的信息：

```

protected void AddUserSet_Click(object sender, EventArgs e)
{
    if (tb.Visible == true)
    {
        tb.Visible = false;
    }
    else
    {
        tb.Visible = true;
    }
}

```

④ btnAdd 控件的 Click 事件，用以将用户信息添加到数据库中：

```
protected void btnAdd_Click(object sender, EventArgs e)
{
    Ma.UserName = this.UserName.Text.Trim();
    Ma.Password =
FormsAuthentication.HashPasswordForStoringInConfigFile(this.UserPwd.Text.Trim(), "MD5");
    Ma.Email = this.UserEmail.Text.Trim();
    Ma.Lever = thisDownListAleave.SelectedValue.Trim();
    if (ba.AddUser(Ma))
    {
        Response.Write("<script language=javascript>alert('添加成功!')</script>");
    }
    tb.Visible = false;
    LoadData();
}
```

⑤ GridView1_RowDataBound()事件，用以控制选中数据行的颜色变化：

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        e.Row.Attributes.Add("onmouseover", "this.style.backgroundColor='#CCCCCC';this.style.color='#FFFFFF';this.style.cursor='#CCCCCC';");
        e.Row.Attributes.Add("onmouseout", "this.style.backgroundColor='#FFFFFF';this.style.color='#FFFFFF';");
    }
}
```

2.10.17 网站后台 修改用户信息页面Admin_EditUser.aspx

网站后台 管理系统用户页面（Admin_AllUsers.aspx）中的 GridView1 控件，其中的修改功能调用了Admin_EditUser.aspx页面。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命名为“Admin_EditUser.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，双击打开“web\adminManager”文件夹下的 Admin_EditUser.aspx 页面，进行页面设计。页面效果如图 2-32 所示。

The screenshot shows a web form titled "修改用户信息" (Modify User Information). The form contains the following elements:

- 用户编号:** A text box containing "[UserCid]".
- 用户名称:** A text box.
- 用户密码:** A text box with a password icon and the text "请输入密码!" (Please enter password!).
- 用户邮箱:** A text box.
- 用户权限:** A dropdown menu with "管理员" (Administrator) selected.
- 确定修改:** A button to confirm the modification.

图 2-32 后台 修改用户信息页面效果图

1. 前台页面设计

网站后台 管理修改用户信息页面Admin_EditUser.aspx，页面设计的具体步骤如下：

首先，将一个表格（Table）控件置于 Admin_EditUser.aspx 页中，为整个页面进行布局。然后，从工具箱中拖动相应控件置于表格中，并在各个控件上右击，打开属性窗口，设置控件的属性。各个控件的属性设置及用途如表 2-30 所示。

表 2-30 控件的属性设置及用途

控 件 类 型	控 件 ID	主要属性设置	用 途
标准/Label 控件	Label1	Text 属性设置为“修 改 用 户 信 息” Font 属性中的 Bold 设置为“True”	说明性描述
	Label2	Text 属性设置为“用户编号： ”	说明性描述
	Label3	Text 属性设置为“用户名称： ”	说明性描述
	Label4	Text 属性设置为“用户密码： ”	说明性描述
	Label5	Text 属性设置为“用户邮箱： ”	说明性描述
	Label6	Text 属性设置为“用户权限： ”	说明性描述
标准/TextBox 控件	UserName	Width 属性设置为“120px”	输入用户名称
	UserPwd	Width 属性设置为“120px”	输入用户密码
	UserEmail	Width 属性设置为“120px”	输入用户邮箱
验证 /RequiredFieldValidator 控 件	RequiredFieldValidator1	ControlToValidate 属性设置为“UserName” ErrorMessage 属性设置为“*用户名不能为空！”	不允许输入的用户 名称为空
	RequiredFieldValidator2	ControlToValidate 属性设置为“UserPwd” ErrorMessage 属性设置为“*用户密码不能为空！”	不允许输入的用户 密码为空
	RequiredFieldValidator3	ControlToValidate 属性设置为“UserEmail” ErrorMessage 属性设置为“*用户邮箱不能为空！”	不允许输入的用户 邮箱为空
验证 /RegularExpressionValidator 控件	RegularExpressionValidator1	ControlToValidate 属性设置为“UserEmail” ErrorMessage 属性设置为“*电子邮箱格式不 正确！” ValidationExpression 属性设置为“\w+([+.'] \w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*”	验证邮箱格式是否 正确
标准/DropDownList 控件	DownListAleave	设置 Items 属性的成员分别为“管理员”和“普 通用户”	用于选取系统管理权 限
标准/Button 控件	Btn_OK	Text 属性设置为“确定修改”	把修改后的用户信 息添加到数据库
标准/ImageButton 控件	ImageButton1	ImageUrl 设置为“~/Web/images/gif026.gif” PostBackUrl 设置为“~/Web/adminManager/ Admin_AllUsers.aspx”	返回 Admin_AllU sers.aspx 页面

2. 后台功能代码

在编辑器页（Admin_EditUser.aspx.cs）编写代码前，需要先定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.UserInfo Ma = new Model.UserInfo();
BLL.Userlogic Ba = new BLL.Userlogic();
```

程序主要代码如下。

① 在 Page_Load 事件中：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Request.QueryString["cid"] == null)
        {
            Response.Write("<script language=javascript>alert('数据库操作出错,请正确操作!')</script>");
        }
        else
        {
            DataBindText(int.Parse(Request.QueryString["cid"].ToString()));
        }
    }
}
```

② 自定义方法 **DataBindText()**，获取要修改的用户信息：

```
public void DataBindText(int id)
{
    DataSet ds = Ba.QueryUserInfoByID(id);
    this.UserCid.Text = ds.Tables[0].Rows[0][0].ToString();
    this.UserName.Text = ds.Tables[0].Rows[0][1].ToString();
    this.UserPwd.Text = ds.Tables[0].Rows[0][2].ToString();
    this.UserEmail.Text = ds.Tables[0].Rows[0][3].ToString();
}
```

③ 按钮 **Btn_OK** 的 Click 事件，将修改后的用户信息添加到数据库：

```
protected void Btn_OK_Click(object sender, EventArgs e)
{
    Ma.U_id = int.Parse(this.UserCid.Text.Trim());
    Ma.UserName = this.UserName.Text.Trim();
    Ma.Password =
FormsAuthentication.HashPasswordForStoringInConfigFile(this.UserPwd.Text.Trim(), "MD5");
    Ma.UserEmail = this.UserEmail.Text.Trim();
    Ma.Lever = this.UserAleave.SelectedValue.Trim();
    if (Ba.UpdateUserInfo(Ma))
    {
        Response.Redirect("Admin_AllUsers.aspx");
    }
}
```

2.10.18 网站后台管理 删除用户信息页面Admin_DeleteUser.aspx

网站后台 管理系统用户页面（Admin_AllUsers.aspx）中的GridView1 控件，其中的删除功能调用了Admin_DeleteUser.aspx页面功能。

在“web\adminManager”文件夹上右击，选择“添加新项”，然后选择“Web 窗体”并命

名为“Admin_DeleteUser.aspx”，不要勾选“选择母版页”。单击“添加”按钮。添加后，只需在编辑器页（Admin_DeleteUser.aspx.cs）编写代码。首先需要定义相关类的对象，以便在编写代码时调用该类中的方法。代码如下：

```
Model.UserInfo Ma = new Model.UserInfo();
BLL.Userlogic Ba = new BLL.Userlogic();
```

程序主要代码如下。
在 Page_Load 事件中，实现删除用户信息的功能：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        if (Request.QueryString["cid"] == null)
        {
            Response.Write("<script language=javascript>alert('数据库操作出错,请正确操作!')</script>");
        }
        else
        {
            Ma.U_id = int.Parse(Request.QueryString["cid"].ToString());
            if (Ba.DeleteAdmin(Ma))
            {
                Response.Redirect("Admin_AllUsers.aspx");
            }
            else
            {
                Response.Write("<script language=javascript>alert('数据库操作出错,请正确操作!')</script>");
            }
        }
    }
}
```

2.10.19 任务小结

本次任务用以实现网站的后台管理功能。通过完成本次任务，要掌握框架的使用，进一步掌握各种控件的属性及应用，熟练掌握数据控件与数据库之间的操作。

2.11 系统的测试

学习目标

- 掌握系统测试的目的
- 掌握系统测试的原则
- 了解常用的系统测试工具
- 建议学时：4 学时

2.11.1 任务名称：系统的测试

本节完成系统的测试。

2.11.2 任务描述

系统测试是为了发现错误而执行程序的过程，成功的测试是发现了至今尚未发现的错误的测试。系统测试可以提高系统的安全性、可靠性、实用性。下面根据本系统的实际情况进行系统测试。

2.11.3 任务分析

完成本系统后对本系统进行系统测试。

2.11.4 系统的测试

1. 系统测试的目的

系统测试是程序的一种执行过程，目的是尽可能发现并改正被测试系统中的错误，提高系统的可靠性。它是系统生命周期中一项非常重要且非常复杂的工作，对系统可靠性保证具有极其重要的意义。在目前形式化方法和程序正确性证明技术还无望成为实用性方法的情况下，系统测试在将来相当一段时间内仍然是系统可靠性保证的有效方法。软件工程的总目标是充分利用有限的人力和物力资源，高效率、高质量地完成系统开发项目。不足的测试势必使系统带着一些未揭露的隐藏错误投入运行，这将意味着更大的危险让用户承担。过度测试则会浪费许多宝贵的资源。到测试后期，即使找到了错误，然而付出了过高的代价。E.W.Dijkstra 的一句名言说明了这一道理：“程序测试只能表明错误的存在，而不能表明错误不存在。”可见，测试是为了使系统中蕴含的缺陷低于某一特定值，使产出、投入比达到最大。

2. 系统测试的基本过程

系统测试是一个极为复杂的过程。一个规范化的系统测试过程通常包括以下基本的测试活动：

- 拟定系统测试计划；
- 编制系统测试大纲；
- 确定系统测试环境；
- 设计和生成测试用例；
- 实施测试；
- 生成系统测试报告。

系统测试需要对整个测试过程进行有效的管理。实际上，系统测试过程与整个系统开发过程基本上是平行进行的，那些认为只有在系统开发完成以后才进行测试的观点是危险的。测试计划早在需求分析阶段即应开始制订，其他相关工作，包括测试大纲的制订、测试数据的生成、测试工具的选择和开发等也应在测试阶段之前进行。充分的准备工作可以有效地克服测试的盲目性，缩短测试周期，提高测试效率，并且起到测试文档与开发文档互查的作用。

系统测试大纲是系统测试的依据，它明确、详尽地规定了在测试中针对系统的每一项功能或特性所必须完成的基本测试项目和测试完成的标准。无论自动测试还是手动测试，都必须满足测试大纲的要求。测试环境是一个确定的、可以明确说明的条件，不同的测试环境可以得出对同一系统的不同测试结果，这正说明了测试并不完全是客观的行为，任何一个测试

的结果都是建立在一定的测试环境之上的。没必要去创造一个尽可能好的测试环境，而只需一个满足要求的、公正一致的、稳定的、可以明确说明的条件。测试环境中最需明确说明的是测试人员的水平，包括专业的、计算机的、经验的能力及与被测程序的关系，这种说明还要在评测人员对评测对象做出判断的权值上有所体现。这一点要求测试机构建立测试人员库并对其参与测试的工作业绩不断做出评价。一般而言，测试用例是指为实施一次测试而向被测系统提供的输入数据、操作或各种环境设置。测试用例控制着系统测试的执行过程，它是对测试大纲中每个测试项目的进一步实例化。

系统测试是保证系统质量和可靠性的关键步骤，是对系统开发过程中的系统分析、系统设计和实施的最后复查。根据测试的概念和目的，在进行信息系统测试时应遵循以下基本原则。

① 应尽早并不断地进行测试。测试不是在应用系统开发完之后才进行的。由于原始问题的复杂性、开发各阶段的多样性及参加人员之间的协调等因素，使得开发各个阶段都有可能出现错误。因此，测试应贯穿在开发的各个阶段，尽早纠正错误，消除隐患。

② 测试工作应该避免由原开发软件的人或小组承担。一方面，开发人员往往不愿否认自己的工作，总认为自己开发的软件没有错误；另一方面，开发人员的错误很难由本人测试出来，很容易根据自己编程的思路来制定测试思路，具有局限性。测试工作应由专门的人员来进行，这样会更客观，更有效。

③ 设计测试方案的时候，不仅要确定输入数据，而且要根据系统功能确定预期的输出结果。将实际输出结果与预期结果相比较就能发现测试对象是否正确。

④ 在设计测试用例时，不仅要设计有效、合理的输入条件，也要包含不合理、失效的输入条件。测试的时候，人们往往习惯按照合理的、正常的情况进行测试，而忽略了对异常、不合理、意想不到的情况进行测试，而这些可能就是隐患。

⑤ 在测试程序时，不仅要检验程序是否做了该做的事，还要检验程序是否做了不该做的事。多余的工作会带来副作用，影响程序的效率，有时会带来潜在的危害或错误。

⑥ 严格按照测试计划进行，避免测试的随意性。测试计划应包括测试内容、进度安排、人员安排、测试环境、测试工具和测试资料等。严格地按照测试计划可以保证进度，使各方面都得以协调进行。

⑦ 妥善保存测试计划、测试用例，作为软件文档的组成部分，为维护提供方便。

⑧ 测试用例都是精心设计出来的，可以为重新测试或追加测试提供方便，可在其基础上修改，然后进行测试。

2.11.5 NUnit测试工具

NUnit 是一个单元测试框架，是专门针对于.NET 写的。NUnit 是 xUnit 家族中的第 4 个主打产品，完全由 C#语言来编写，并且编写时充分利用了许多.NET 的特性反射、客户属性等。最重要的一点是，它适合于所有.NET 语言。

在 NUnit 面板的中间可以看到测试的进度条(或者叫状态条)，这里会有 3 种不同的信号：绿色表示所有的测试用例都通过；红色表示测试用例中有失败；黄色表示有些测试用例忽略，但测试过的没有失败。在进度条的上方会有一些统计信息，它们所表示的意义如下。

➤ **Test Cases:** 表示加载的所有测试用例的个数。

➤ **Tests Run:** 表示已经运行的测试用例个数。

➤ **Failures:** 表示到目前为止运行失败的测试用例个数。

- Ignored: 表示忽略的测试用例个数。
- Run Time: 表示运行所有测试用例所花费的时间。
- NUnit 框架是基于 Attribute 的, 这和 VSTS 是一致的, 但它们之间所使用的 Attribute 并不相同。现在编写一个简单的 NUnit 测试示例, 如有下面这样一段代码:

```
1: public class Calculator
2: {
3:     public int Add(int a, int b)
4:     {
5:         return a + b;
6:     }
7: }
```

现在对 Add 方法编写单元测试, 在开始之前, 需要添加对 `nunit.framework` 的引用。NUnit 中用到的 Attribute 都定义在该程序集中, 在 `CalculatorTest` 中引入命名空间:

```
1: using NUnit.Framework;
```

编写测试类, 在 NUnit 中每个测试类都必须加上 `TestFixture` 特性, 代码如下所示:

```
1: [TestFixture]
2: public class CalculatorTest
3: {
4:
5: }
```

现在编写 `TestAdd` 测试函数, NUnit 中每个测试函数都需要加上 `Test` 特性, 如下代码所示。这里添加了两个断言, 一是假定创建的对象不为空, 二是测试 `Add` 方法是否返回预期的结果:

```
1: [Test]
2: public void TestAdd()
3: {
4:     Calculator cal = new Calculator();
5:     Assert.IsNotNull(cal);
6:     int expectedResult = 5;
7:     int actualResult = cal.Add(2,3);
8:     Assert.AreEqual(expectedResult, actualResult);
9: }
```

至此, 一个完整的测试用例编写完成, 使用 NUnit 可视化工具打开该程序集后, 单击 **Run** 按钮, 全是绿灯表示测试通过。

2.11.6 任务小结

通过本次任务, 应该明确系统测试的目的和意义, 了解测试工具的一般用法。

2.11.7 练习题

1. 系统测试的意义是什么?
2. 如何使用 NUnit 进行类的测试?

第 3 章 企业在线客服信息管理系统

(AJAX技术应用)

3.1 用户需求的分析与处理

学习目标

- 理解用户需求
- 根据用户需求建模
- 撰写规格说明书
- 建议学时：8 学时

随着计算机网络的进一步普及与应用，各种类型的电子商务网站也应运而生，从门户网站到网络购物，可谓种类繁多、应有尽有。其应用人群从青少年到中老年，覆盖范围也较广。由于各个文化层次的人员都有，难免在使用上产生各种各样的疑问，如不知道如何下订单或者不能快速找到想要的产品，等等。如果在这个时候能够有一个专业的客户服务人员加以指导，往往会给客户以满意的感受。然而，网站用来解决此类问题无非是电子邮件或电话咨询及留言板、论坛等，这些渠道要么消耗金钱，要么消耗时间，其效率都不是特别高，用户满意度也比较低。随着在线客服技术的大规模应用，使得原来需要很长时间等待的过程变得简单化，也使得原来需要专业人士解答的专业问题的复杂化得到解决。因此，现在有许多有规模、有实力的专业网站在解决此类问题时选用的都是在线客服答疑这类服务。使用 ASP.NET 技术解决在线客服的设计方案也常被大的软件公司所采用。

在线客服的功能要求是用户与客服经理采用一对一的方式进行交流，即可以在线问答，也可以给客服经理发送离线留言。在一对一的在线交流过程中，需要聊天记录自动刷新，而该页面的刷新往往会造成整个页面的刷新，这样的操作通常会消耗远程服务器巨大的系统资源。如果这是一个规模庞大的网站，当成百上千的用户同时登录访问时，服务器会因无法满足多用户的请求而死机。这样的问题并不能阻止 ASP.NET 技术在在线客服方面的应用，其中 AJAX 技术的加盟使得利用 ASP.NET 技术开发具备实时异步刷新功能的在线客服变得简单而快捷。

3.1.1 任务名称：用户需求的分析与处理

3.1.2 任务描述

本例用户是蓝星酒店的业务主管，目前酒店的各项业务介绍都是采用电话问答的方式进行的。随着网络的迅猛发展，手工方式已经无法满足越来越多用户的需要，酒店负责人决定

启用一套新的在线客服软件系统，并要求满足以下条件：

- ① 系统应提供统一的客服与客户交流的界面。
- ② 客户与客服交流之前应进行登记。
- ③ 客户可以自行选择客服人员进行交流。
- ④ 客户可以对离线的客服人员提出问题。
- ⑤ 客户与客服直接的交流方式仅限于文字交流。
- ⑥ 客户与客服的交流内容应能长久保存下来。
- ⑦ 客户与客服的交流内容除了保存基本的对话信息外，还应保存发消息人的姓名、发消息的时间、日期等信息。
- ⑧ 该在线客服系统还应提供所有交流信息的预览功能，并提供消息的管理功能。
- ⑨ 该系统应提供客服人员的管理功能。

在线客服管理系统就是要满足以上功能的软件。需求就是一套软件的最终目标。但并不是所有的功能都能达到，可能还存在某些功能的不可达到性，需要进一步的分析和论证。

3.1.3 任务分析：需求分析人员分析用户的需求

当软件开发公司接到用户的开发需求时，并不是按照最初的需求进行程序开发的。这是因为，作为软件的需求者——客户而言，其所处的行业各不相同，各自的知识结构也比较复杂，并不是每一个客户都可以清晰地描述自己的实际需求，往往都是一些业务骨干凭借平时的业务流程进行需求描述，这就可能造成需求描述不统一的无奈结局。往往许多软件开发项目在需求分析的初期由于没有做到充分了解客户的实际需要，而匆忙开发，中途又遇到无法完成的功能需要，最终不得不选择放弃项目，导致产生违约。因此，一个成熟的软件开发团队通常需要组建一个完善的系统分析及需求分析部门，通过这些前期服务人员不断与软件需求客户取得各种联络，不断完善客户的需求与目的并最终反馈到项目的开发部门，只有这样开发出来的产品才可能是客户最终想要的产品。事实上大多数从事非软件开发方面的产品需求客户，到最后连自己都无法 100% 准确地描述出到底需要一个什么样的系统，这也就是为什么软件开发团队花费了巨大精力，而当他们把产品提交给客户时，客户却认为这并不是他们最终想要的产品的道理。

通过需求分析人员与产品需求客户的进一步交流，最终的谈话结果以文档的方式提交到系统分析人员手中，再由系统分析人员得出结论，并通过需求分析人员转交到产品需求客户那里，这样不断反复数次，最终敲定以下功能开发方案。最后还需要签订开发协议，以免产品需求客户提出新的功能方案而使得软件开发过程产生可能违约的状况。

- ① 客户可以在网上直接提问，提问的方式为文字。
- ② 客户在提问之前需要先登录系统，以辨别客户的身份。当客户名称不在登录许可的范围内时，要给出注册新用户的界面，以增加客户人员。
- ③ 客户可以对离线的客服人员提问，离线消息需要长期保存。
- ④ 客户可以对在线的客服人员提问。
- ⑤ 客户可以随意选择在线或离线的某一个客服人员进行提问交流。
- ⑥ 系统自动保存客户提问的时间、日期及提问者的用户名。
- ⑦ 客服人员在进入工作台状态之前要先登录。客服的工作台就是客服与客户进行交流答疑的网页界面。

- ⑧ 只有对该客服提问的客户才可以显示在客服工作台界面中。
- ⑨ 客服可以看到客户对于该客服离线时发出的提问消息。
- ⑩ 客服可以看到客户发出消息的时间、日期、提问人的用户名等信息。
- ⑪ 客服也可以看到自己发出消息的时间、日期等信息。
- ⑫ 整个在线客服系统、客服和客户双方的交谈留言全部需要保存下来，以备查看。
- ⑬ 在线客服系统应提供客服的管理方案，具体为添加客服等功能。
- ⑭ 在线客服系统应提供客服与客户交谈信息的预览功能及管理功能如删除非法信息的功能。
- ⑮ 保存消息的发出者和接收者。

3.1.4 需求建模

一个大的软件系统是为了解决一系列的业务需要而编写的，其中很多功能是相互联系的，如果对其进行有效的整理与分析，将其中相互关联的功能作为一个功能模块来进行开发，可以极大地减少错误的产生，并提高产品的开发效率。

实际上作为一个在线客服系统而言，它所提供的功能只有以下几类：

- ① 提供一个统一的操作界面，供所有登录上来的用户进行文字交流。
- ② 在交流过程中所有的文字信息都要予以保存，以备核对。
- ③ 所有的聊天记录都有发送时间、日期、发送人等信息。
- ④ 用户可以选择其中一个客服人员进行交流。
- ⑤ 在客服管理系统的后台部分，需要提供聊天信息的查阅功能，并提供有效的管理功能。
- ⑥ 在客服管理系统的后台部分，需要提供客服人员的添加功能。
- ⑦ 所有提问的用户在使用在线客服系统之前都需要现行登录。

有了以上的一些了解，再加上一个通用的在线客服系统所具备的功能，大致可以把蓝星酒店在线客服系统的功能划分为以下模块。

- 人员管理模块：主要包括注册普通用户、注册客服人员、普通用户登录、客服人员登录、判断客服人员是否离线、获取所有客服、获取所有提问用户等功能，如图 3-1 所示。

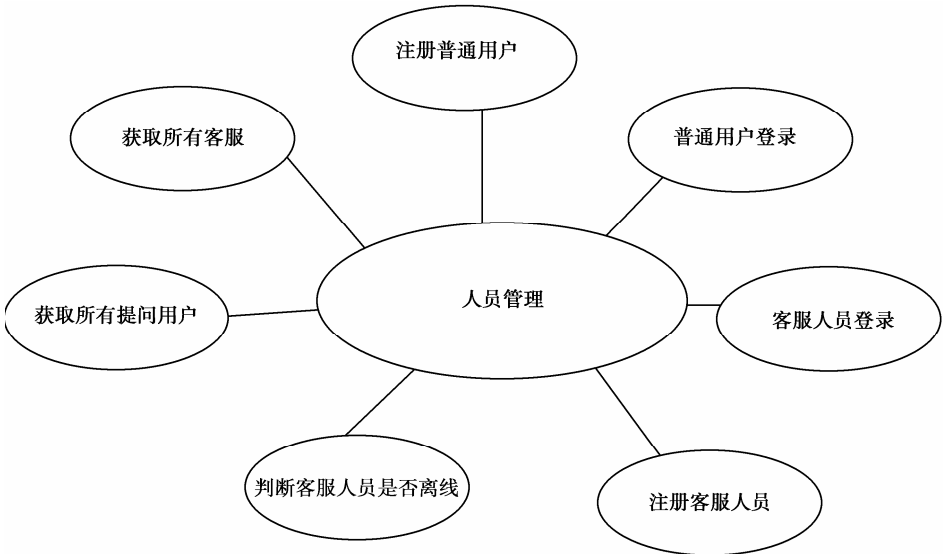


图 3-1 人员管理模块

➤ 消息管理模块：主要包括发送离线消息、获取针对某客服的消息、获取所有消息、发送在线消息等功能，如图 3-2 所示。

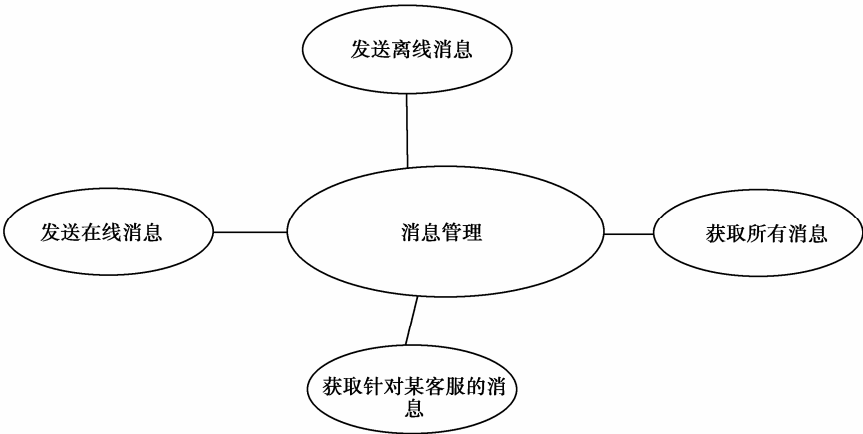


图 3-2 消息管理模块

3.1.5 撰写规格说明书

需求规格说明阐述一个软件系统必须提供的功能和性能及其所要考虑的限制条件，它不仅是系统测试和用户文档的基础，也是所有子系列项目规划、设计和编码的基础。它应该尽可能完整地描述系统预期的外部行为 and 用户可视化行为。

可采用需求规格说明书模板在组织中为编写软件需求文档定义一种标准模板，该模板为记录功能需求和各种其他与需求相关的重要信息提供了统一的结构。许多组织一开始都采用 IEEE 标准 830—1998（IEEE 1998）描述的需求规格说明书模板，如表 3-1 所示。

表 3-1 IEEE 1998 描述的需求规格说明书模板

	1	2	3	4	5	6
A. 引言	目的	文档约定	预期的读者和阅读建议	产品的范围	参考文献	
B. 综合描述	产品的前景	产品的功能	用户类和特征	运行环境	设计和实现上的限制	假设和依赖、附录
C. 外部接口需求、附录	用户界面、附录	硬件接口	软件接口	通信接口		
D. 系统特性	说明和优先级	激励/响应序列	功能需求			
E. 其他非功能需求	性能需求	安全设施需求	安全性需求	软件质量属性	业务规则	用户文档
F. 其他需求						
G. 附件	词汇表	分析模型	待确定问题的列表			

编辑一张在软件需求规格说明中待确定问题的列表，其中每个表项都编上号，以便于跟踪调查。

① 指明需求来源：指明需求的来源为了让所有的项目风险承担者明白需求规格说明书中为何提供这些功能需求。要能追溯每项需求的来源，这可能是一种使用实例或其他的客户要求，也可能是某项更高层的系统需求、业务规范、政府法规、标准或别的外部来源。

② 为每项需求注上标号：为了满足软件需求规格说明的可跟踪性和可修改性的质量标准，必须唯一确定每个软件需求。为每项需求注上标号、制定一种惯例，来为需求规格说明书中的每项需求提供一个独立的、可识别的标号或记号。

③ 记录业务规范：是指关于产品的操作原则，如谁能在什么情况下采取什么动作。将这些编写成需求规格说明书中的一个独立部分，或一个独立的业务规范文档。

④ 需求跟踪能力矩阵：建立一个矩阵，把每项需求与实现、测试它的设计和代码部分联系起来。这种需求跟踪能力矩阵同时也把功能需求和高层的需求及其他相关需求联系起来。在开发过程中建立这个矩阵，不要等到最后才去补建。

这里还要介绍需求规格说明书中设计阶段用到的图形模型——数据字典、数据流图、实体联系图、状态转换图、对话图和类图。

- 数据字典：一个定义应用程序中使用的所有数据元素和结构的含义、类型、数据大小、格式、度量单位、精度及允许取值范围的共享仓库。数据字典的维护独立于软件需求规格说明，并且在产品的开发和维护的任何阶段，各个风险承担者都可以访问数据字典。它定义了原数据元素、组成结构体的复杂数据元素、重复的数据项、一个数据项的枚举值及可选的数据项。
- 数据流图：是结构化系统分析的基本工具。一个数据流图确定了系统的转化过程、系统所操纵的数据或物质的收集（存储），还有过程、存储、外部世界之间的数据流或物质流。数据流模型把层次分解方法运用到系统分析上，这种方法很适用于事务处理系统和其他功能密集型应用程序。
- 实体联系图：描绘了系统的数据关系。分析实体联系图有助于对业务或系统数据组成的理解和交互，并暗示产品将有必要包含一个数据库。
- 状态转换图：实时系统和过程控制应用程序可以在任何给定的时间内以有限的状态存在，当满足所定义的标准时，状态就会发生改变，如在特定条件下，接收到一个特定的输入激励。
- 对话图：在许多应用程序中，用户界面可以看做是一个有限状态机，在任何情况下仅有一个对话元素（如一个菜单、工作区、行提示符或对话框）对用户输入是可用的。在激活的输入区中，用户根据其所采取的活动，可以导航到有限个其他对话元素。因此，许多用户界面可以用状态转换图中一种称为对话图的来建模。对话图描绘了系统中的对话元素和它们之间的导航连接，但它没有揭示具体的屏幕设计。
- 类图：面向对象的软件开发优于结构化分析和设计，它运用于许多项目的设计中，从而产生了面向对象分析、设计和编程的域。类图是用图形方式叙述面向对象分析所确定的类以及它们之间的关系。

3.1.6 任务小结

一个项目得以最终完成，在很大程度上决定于它是否经历过详细而周密的用户需求调研。只有真正满足用户的需求，才可以称得上是成功的软件。

3.1.7 练习题

结合自身的认识，写一篇关于在线客服系统的需求分析报告。

3.2 项目计划安排

学习目标

- 掌握 Project 2003 制作甘特图的方法
- 建议学时：8 学时

3.2.1 任务名称：项目计划安排

3.2.2 任务描述

一个软件开发团队要想有效率地协同开发同一个项目，除了需要具备熟练的软件开发经验以外，还需要制订周密的发展计划并严格执行。对于项目规划软件，本例使用 Microsoft 提供的 Project 2003，通过其制定甘特图规划项目计划安排。

3.2.3 任务分析

项目计划安排主要说明在某一段时间由哪些人、使用哪些设备、完成哪项任务、达到哪些目的等。

3.2.4 创建甘特图

Project 2003 是一个可视化的项目管理软件，下面利用它来创建一个开发进度安排表。打开 Project 2003，出现如图 3-3 所示界面。用鼠标单击方框处，可以填写任务名称、工期、开始时间、完成时间、前置任务等信息。

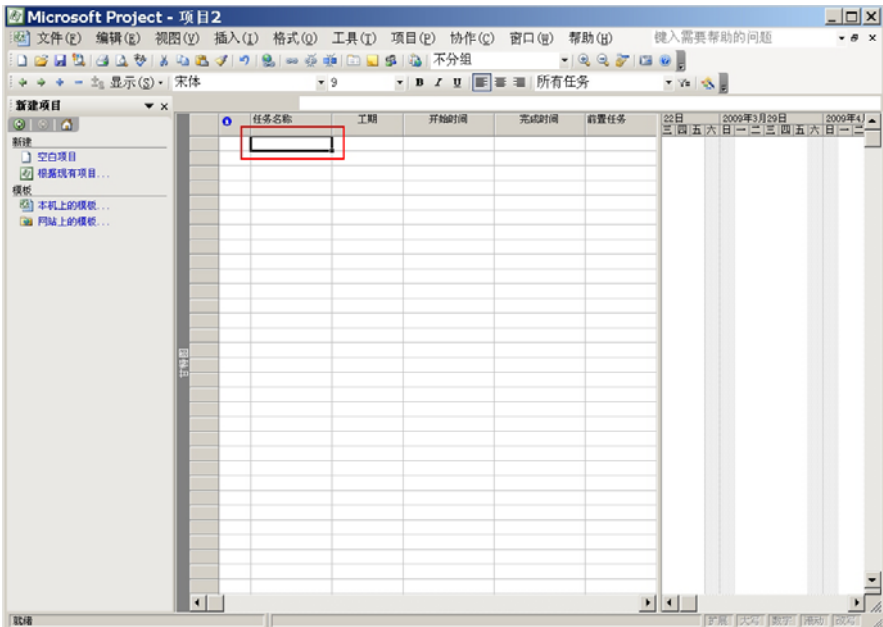


图 3-3 Project 2003 工作界面

逐项填写后，得到内容如图 3-4 所示，选取“文件”→“另存为”，指定一个名称后保存，然后打印出若干份，分发到开发部门人员手中。

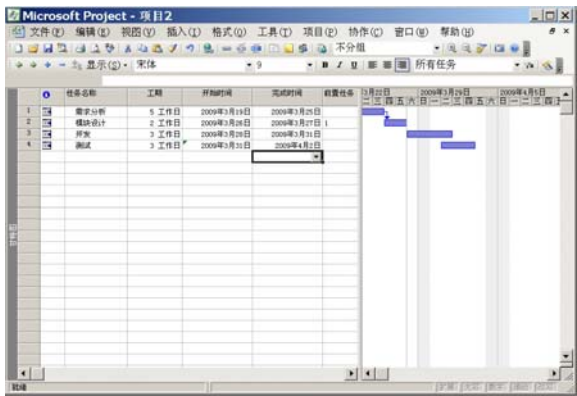


图 3-4 甘特图制作界面

3.2.5 实现项目的资源

一个项目必须包括开发人员和开发设备。在项目进度中所标明的资源就是完成该项目的人员和设备。各个开发阶段所需要的参与人员和设备各不相同。

在 Project 2003 中单击“视图”→“资源工作表”，打开如图 3-5 所示制定项目资源界面。

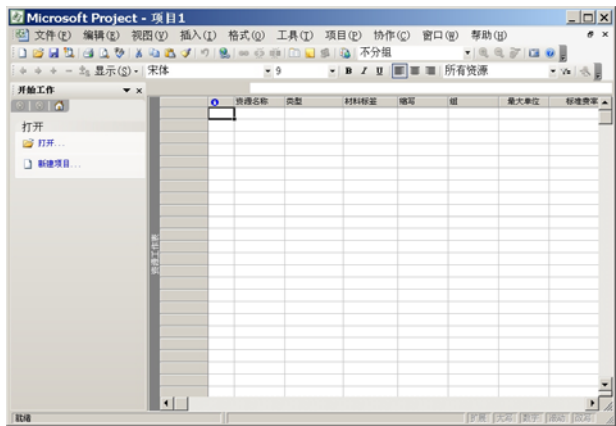


图 3-5 制定项目资源界面

双击表格中的任意方格，可打开“资源信息”窗口，如图 3-6 所示。

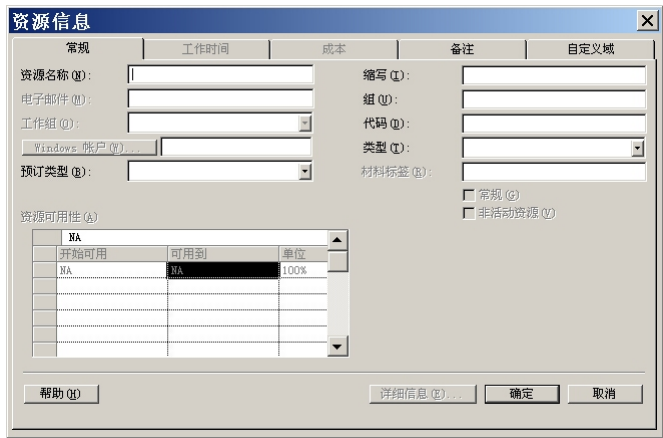


图 3-6 “资源信息”窗口

安排好开发人员和开发设备后，将该页面打印出来分发给设备提供部门和人力资源部门，调派开发人员，组织培训，组织考试。人员调配完毕后，进入下一阶段。

3.2.6 编写前期的项目计划表

由于在做项目进度安排时，无法预知详细的软件开发模块，只是知道具体需要哪些功能，因此开发前期无法详细描述开发进度，只能给出一个开发的时间段安排，由系统分析人员做完具体的模块设计后，再在项目计划安排中予以补充。

3.2.7 任务小结

掌握利用 Project 2003 等项目规划软件制订项目计划安排，防止团队开发过程中产生个体理解差异，进而影响后续开发的问题。

3.2.8 练习题

用 Project 2003 为在线客服管理系统制定完整的项目计划安排。

3.3 系统架构设计

学习目标

- 理解系统架构设计的概念
- 建议学时：4 学时

3.3.1 任务名称：系统架构设计

3.3.2 任务描述

系统架构是一个软件系统从整体到部分的最高层次的划分。系统架构设计除了要实现用户要求的功能外，还应实现以下目标。

- ① 可扩展性：满足用户对新增功能的需求。
- ② 安全性：保护用户的数据是一个系统必须具备的攻能。
- ③ 可定制性：允许根据不同用户的需求进行调整。
- ④ 稳定性：软件运行时保持稳定的性能，才可以提高用户的工作效率。

3.3.3 任务分析

如同软件本身有其要达到的目标一样，架构设计也具有要达到的目标。一般而言，软件架构设计要达到如下的目标：

- 可靠性（Reliable）；
- 安全性（Secure）；
- 可扩展性（Scalable）；
- 可定制化（Customizable）；
- 可扩展性（Extensible）；

- 可维护性 (Maintainable);
- 客户体验 (Customer Experience);
- 市场时机 (Time to Market)。

3.3.4 任务小结

时下流行的软件开发模式为面向对象开发，本软件运行平台基于互联网的远程服务器，因此，本系统的构架基线拟定为三层构架，即表示层、数据访问层、控制层。这样既有利于系统升级，也有利于后期维护等操作。

目前大多数采用 ASP.NET 技术开发的网站都是基于重用类库 DLL 项目的方式扩展网站功能的。通过引用外部类库 DLL 项目，可以极大地缩短项目开发周期，降低代码维护成本；但其也有一定缺点，就是在编写引用外部类库的过程中，一旦涉及对外部类库源代码修改的操作，都要求重新对类库编译、生成，进而重新在当前项目中进行新的引用，如此反复，在开发过程中难免有所遗漏。这样的错误往往发生在初学者阶段，而且不易被发觉，给维护代码带来很多麻烦。因此，在本系统的介绍过程中，采用文件夹组织类文件的方式，对需要分层的代码使用物理上的分层，这样做的好处是，免去了每次修改外层类代码所产生的重新生成、重新编译、重新引用的操作，进一步提高了编程效率。

3.3.5 练习题

查阅图书、网络等资料，说明网络项目开发中所说的三层架构设计主要是指哪三层？

3.4 模块的详细设计

学习目标

- 掌握 Visio 2003 软件绘制类图的方法
- 掌握 Visio 2003 绘制用例图、顺序图的方法
- 建议学时：16 学时

3.4.1 任务名称：模块的详细设计

3.4.2 任务描述

前面的介绍中把在线客服系统划分为两大功能模块：人员管理、消息管理。由此便产生两个实体类：User（用户类），Message（消息类）。整个系统初定项目名称为 OnLine，因此类名称使用“OL”+“下画线”+“类名”的命名方式。同时，还有两个数据层访问类——UserDAL 类和 Message 类，两个管理类——UserBLL 类和 MessageBLL 类。以下简要介绍两个位于 Model 层的实体类的规格说明，其他类的设计在后面再详细介绍。

主要需完成以下内容：

- 类的类别及规格说明；
- 类图的设计；
- 用例的基本事件流；

- 扩展事件流;
- 异常事件流;
- 绘制用例图;
- 绘制顺序图。

3.4.3 任务分析

根据前面的用户需求分析、项目计划安排、系统概念设计，将用户实际业务逻辑在 Visio 2003 软件的支持下构建成图形化的各种功能叙述，准确传达系统分析员及软件架构师的设计理念。

3.4.4 类的列表及规格说明

1. User类（见表 3-2）

表 3-2 User 类的设计

属 性 名 称	数 据 类 型	说 明
Name	String	人员名称
Password	String	登录密码
OnLine	String	在线状态，取值范围“在线/离线”
isAdmin	String	是否为管理员，取值范围“普通用户/客服”

2. Message类（见表 3-3）

表 3-3 Message 类的设计

属 性 名 称	数 据 类 型	说 明
fromUser	String	消息发送者
Msg	String	消息
ToUser	String	消息接收者
dateTime	String	消息发出时间
Online	String	消息发出时发送方是何状态，取值范围“在线/离线”

3.4.5 用图例实现设计

1. User类（见图 3-7）

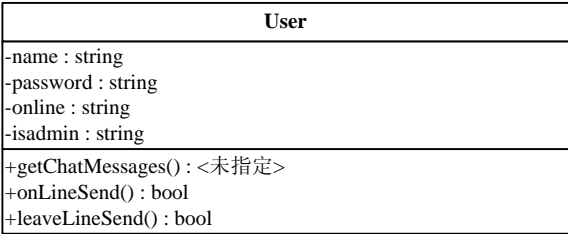


图 3-7 User 类图

2. Message类（见图 3-8）

Message
-FromUser : string
-Msg : string
-ToUser : string
-DateTime : string
-Online : string
+Login() : bool
+Reg() : bool
+UpdateIsAdmin() : bool
+ExistUser() : bool
+getUserList() : <未指定>
+isAdmin() : string
+UpDateOnLine() : bool
+OnLine() : bool

图 3-8 Message 类图

3.4.6 用例的基本事件流、扩展事件流、异常事件流

1. 用户操作部分

- 用户进入在线客服系统界面；
- 进行登录验证；
- 登录成功，进入在线客服咨询界面；
- 系统自动获取客服列表；
- 显示客服在线状态；
- 系统自动获取用户在该系统所发出的消息；
- 系统自动获取客服人员对该用户的所有消息；
- 用户选择客服人员，书写消息内容，发出消息。

2. 客服操作部分

- 客服进入在线客服后台管理系统界面；
- 进行登录验证；
- 登录成功后，显示所有自己曾经发出的消息；
- 系统自动获取用户对该客服的离线消息；
- 系统自动获取用户对该客服的在线咨询消息；
- 系统自动获取当前对该客服发出消息的用户名称列表；
- 客服人员输入消息，发出消息等。

3.4.7 用例的顺序图与活动图

由于涉及用例较多，篇幅又有限，因此仅以第一个用例——用户登录为例进行介绍。用户登录验证顺序图如图 3-9 所示。

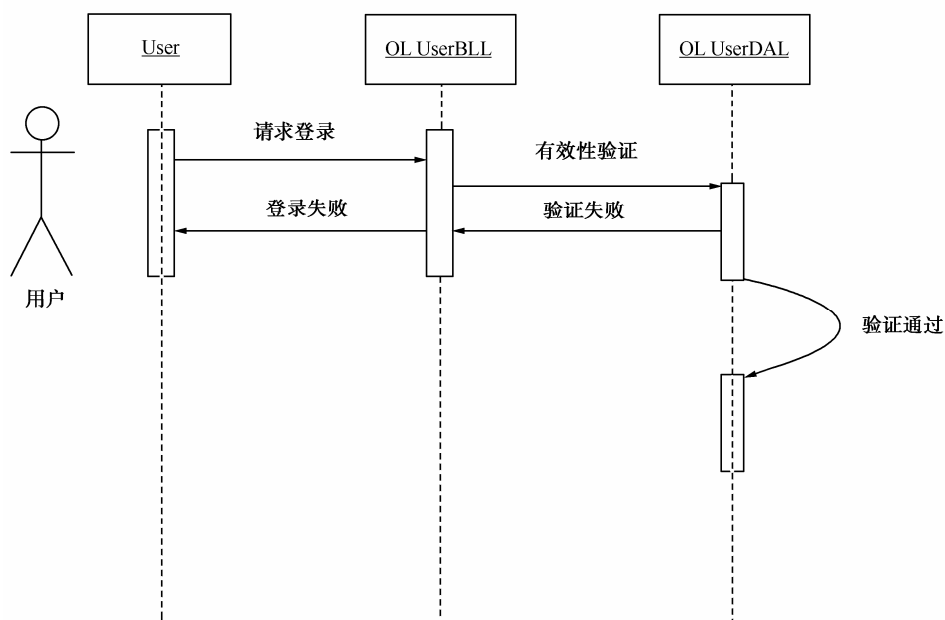


图 3-9 用户登录验证顺序图

用户登录验证活动图如图 3-10 所示。

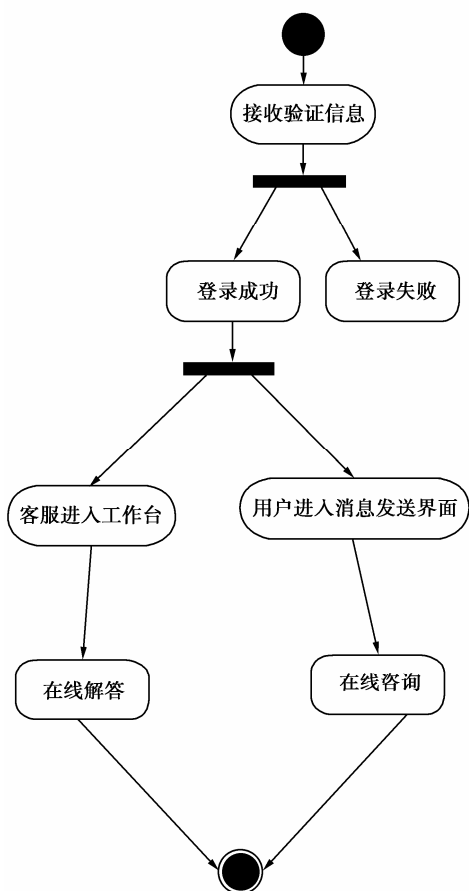


图 3-10 用户登录验证活动图

3.4.8 任务小结

以上各类图形的设计与安排离不开前期的用户需求分析，在制定各类图形的设计方案的同时，应不断地与客户代表进行交流，不断改进设计方案，防止出现“答不对题”的现象。

3.4.9 练习题

用自己的理解描述 3.4.7 节中的用例顺序图所表达的含义。

3.5 数据库设计

学习目标

- 掌握使用 SQL Server 2005 设计数据库的方法
- 掌握使用 SQL Server 2005 设计表的方法
- 建议学时：4 学时

3.5.1 任务名称：数据库设计

3.5.2 任务描述

根据前面介绍的 User 类及 Message 类所涉及的数据项，设计数据库及表，定义各个字段的数据类型及字段名称。

3.5.3 任务分析

本在线客服系统采用 SQL Server 2005 数据库，数据库命名为 OnLine，共有两个表，分别为 OL_User 表和 OL_Message 表。其中 OL_User 表包含 Name、Password、OnLine、IsAdmin 字段，OL_Message 表包含 FromUser、Msg、toUser、DateTime、OnLine 字段。

OL_User 表的设计如表 3-4 所示。

表 3-4 OL_User 表的设计

Name	Varchar(50)	用户名称
Password	Varchar(50)	登录密码
OnLine	Varchar(50)	在线状态
IsAdmin	Varchar(50)	用户类型

OL_Message 表的设计如表 3-5 所示。

表 3-5 OL_Message 表的设计

FromUser	Varchar(50)	消息发送者
Msg	Varchar(50)	消息
toUser	Varchar(50)	消息接收者
DateTime	datetime	消息日期
OnLine	Varchar(50)	接收方状态

3.5.4 生成数据库

生成数据库的方式多种多样，既可以用数据库生成工具，也可以用 SQL Server 2005 自身提供的功能进行数据库的生成。

打开 SQL Server 2005，在其工作界面左侧的“数据库”上右击鼠标，在弹出的快捷菜单中选择“新建数据库”，如图 3-11 所示。

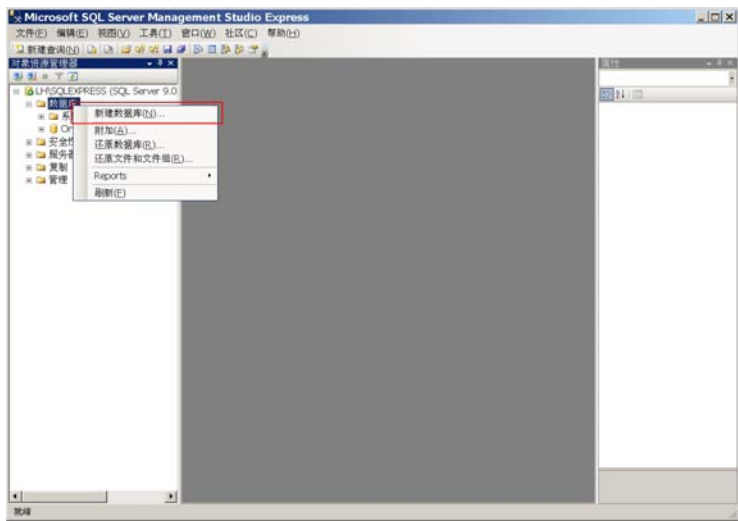


图 3-11 新建数据库

打开“新建数据库”窗口，在“数据库名称”处输入“OnLine”，单击“确定”按钮，如图 3-12 所示。

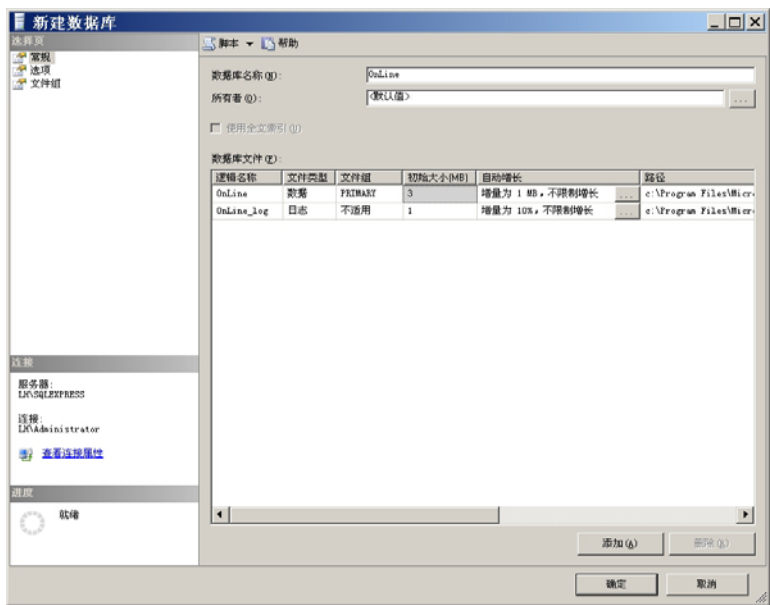


图 3-12 定义数据库名称

鼠标右键单击刚刚创建的“OnLine”数据库，在弹出的快捷菜单中选择“新建表”，如图 3-13 所示。

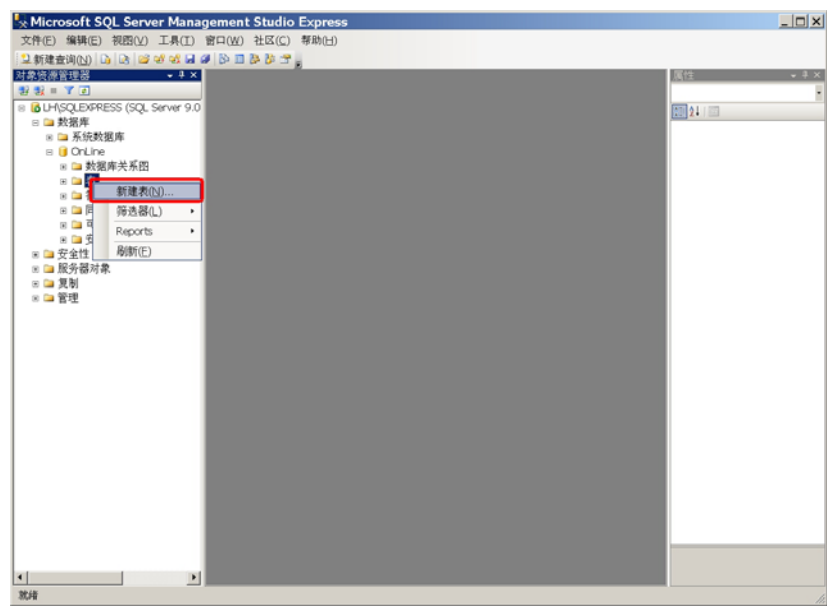


图 3-13 新建表

录入如图 3-14 所示信息，单击“保存”图标，输入表名称“OL_Message”。

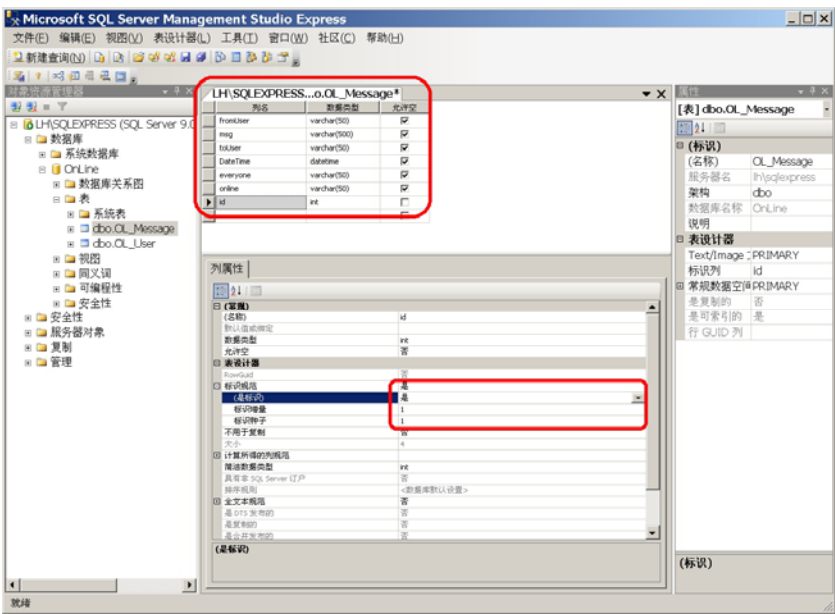


图 3-14 定义 OL_Message 表结构

重复以上操作：建立表“OL_User”，如图 3-15 所示。

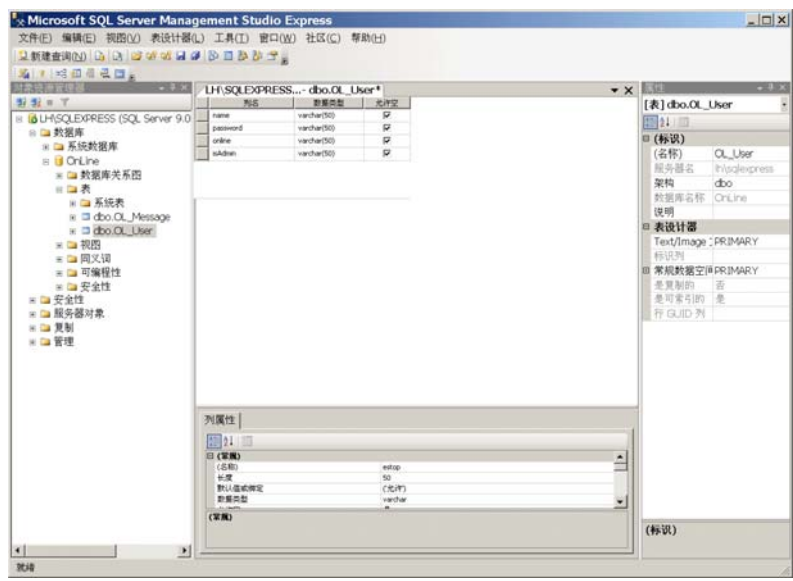


图 3-15 建立 OL_User 表

3.5.5 任务小结

数据库设计既可以使用代码方式，也可以使用向导方式。通常向导方式轻松快捷，但掌握一定的代码编写能力对将来涉及更多的数据检索方面的项目有很大帮助。

3.5.6 练习题

尝试使用 SQL Server 2005 设计一个数据库，要求具有 3 个表，包含姓名、班级、学号、宿舍号、年龄、性别、英语成绩、数学成绩、设定各字段所属表的设计。

3.6 界面设计

学习目标

- 掌握母版页设计方法
- 掌握调整编码模式下的字体设置
- 建议学时：4 学时

Windows 操作系统之所以能够取得巨大的成功，不仅是因为其拥有强大的功能，更重要的是它采用了极其灵活、方便的人机交互界面。合理的界面布局，既体现了功能的易用性又增加了美观性。因此任何一个项目都应该有其独特的界面设计，一个网站的成功与否，除了关键的功能外，界面设计也是重要的评判标准。

3.6.1 任务名称：界面设计

3.6.2 任务描述

在线客服系统对于普通用户部分只有登录页面和客服咨询页面可以选择，因此不采用母版页的应用方式。而作为在线客服后台管理部分则包含多个功能模块，这里设计一个母版页统一所有后台管理页的界面风格。

3.6.3 任务分析

在线客服系统大致分为登录界面、用户与客服交流界面、在线客服系统后台管理界面，其中在线客服系统后台管理界面又包含若干功能界面。因此登录界面和用户与客服交流界面仅各自设计一个界面即可，而在线客服系统后台管理界面则应具有统一的风格与样式，对在线客服系统后台管理界面进行母版页设计是本次任务的核心。

3.6.4 前期准备

1. 创建网站项目，添加母版页

① 打开 VS2008，选择“文件”→“新建”→“网站”，新建网站，命名为“OnLine”如图 3-16 所示，单击“确定”按钮保存。

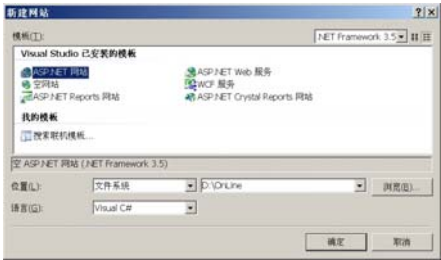


图 3-16 新建网站

② 按 F5 键，VS2008 系统提示是否执行调试功能，选择“是”，然后关闭浏览器，此时 VS2008 系统自动在网站根目录下创建“web.config”文件。

③ 在“解决方案资源管理器”中鼠标右键单击项目名称，在弹出的快捷菜单中选择“添加新项”，如图 3-17 所示。

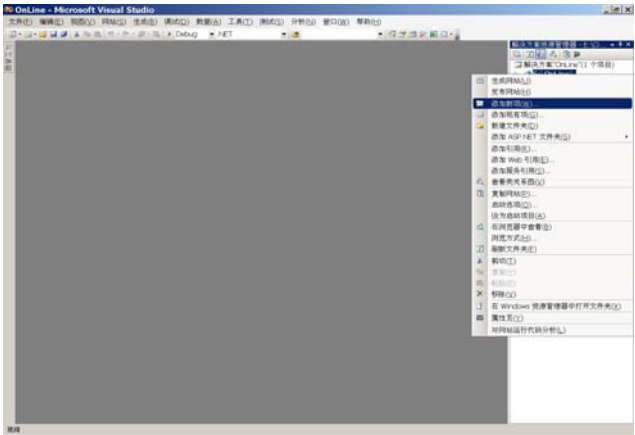


图 3-17 添加新项

④ 在打开的“添加新项”窗口中选择“母版页”，输入文件名“adminMasterPage.master”，如图 3-18 所示，单击“添加”按钮。

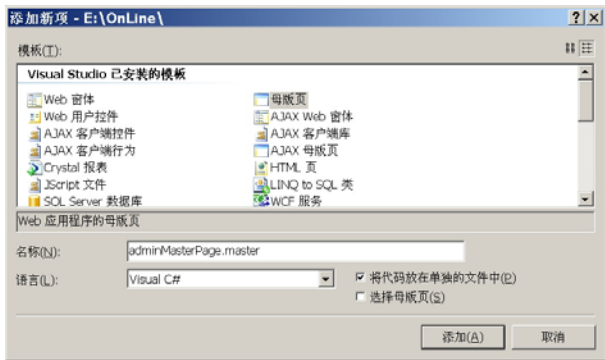


图 3-18 添加母版页

2. 母版页HTML源代码

母版页的 HTML 源代码如下：

```
...
<head runat="server">
    <title>无标题页</title><asp:ContentPlaceHolderid="head" runat="server"></asp:ContentPlaceHolder>
</head>
<body><form id="form1" runat="server">
    <div style="text-align:center" >
<tablewidth="889"height="647"style="border-style:solid;border-color:#6699FF;
background-image:urlimg/admin.jpg)">
    <tr style="height:60px; width:700px; ">
        <td colspan="2"><divalign="center"style="font-size:36px;">Ajax 在线客服管理中心</div></td>
    </tr><tr >
        <td width="163" height="500"><div >
            <p><a href="admingzt.aspx">工作台</a></p>
            <p><a href="adminkfgl.aspx">后台客服管理</a></p>
            <p><a href="adminxxgl.aspx">后台消息管理</a></p></div></td>
        <td width="708" ><asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
            <asp:ContentPlaceHolder></td></tr></table></div></form></body>
</html>
```

在项目开发过程中，如果感觉代码字号太小，可以适当调整源代码视图下的字号。

单击“工具”→“选项”，如图 3-19 所示。

打开“选项”对话框，找到“字体和颜色”，如图 3-20 所示，在这里可以随意修改代码的字体、字号、颜色、背景色等。在左侧的树形结构里还可以进一步对整个 VS2008 开发工具做更多的个性化设置。

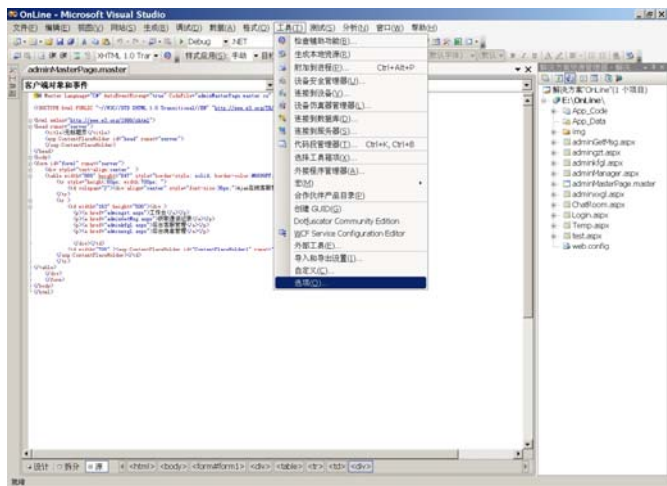


图 3-19 调整字号大小

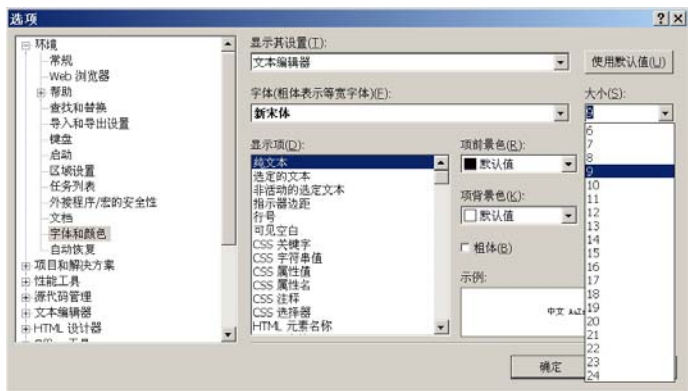


图 3-20 调整字体和颜色

3. 添加网站图片文件夹及背景图片

在前面的母版页 HTML 源代码中可以看到一个关于背景图片的设置 `background-image:url (img/admin.jpg)`，这里需要在“解决方案资源管理器”下的网站项目根目录里添加一个文件夹，命名为 `img`。在这个文件夹上右击鼠标，在弹出的快捷菜单中选择添加“现有项”，然后添加 `admin.jpg` 图片。

在前面的母版页 HTML 源代码中还可以看到如下几行代码，它们表示后台管理系统包含的 3 个主要功能及 3 个功能页面。由于这 3 个页面还没有创建，所以可以先这样写上，但并不运行，主要起到占位的作用。

```
<p><a href="admingzt.aspx">工作台</a></p>
<p><a href="adminkfgl.aspx">后台客服管理</a></p>
<p><a href="adminxxgl.aspx">后台消息管理</a></p>
```

母版页最后的效果如图 3-21 所示。



图 3-21 母版页最后效果图

3.6.5 相关技能与知识

本任务主要练习使用母版页进行网站页面的统一布局，在母版页 HTML 源代码模式下可以看到两组标签，分别为：

```
<asp:ContentPlaceHolder id="head" runat="server"></asp:ContentPlaceHolder>  
<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server"></asp:ContentPlaceHolder>
```

在设计母版页时，添加在这两组标签以外的内容会在采用母版页的其他 Web 窗体中显现出来。设计母版页时应注意这一点。

3.6.6 任务小结

母版页的合理运用可以极大地降低网站开发的工作量，并提供较高的网站维护速度。一个优秀的网站离不开优秀的网站设计风格，而母版页的出现使得网站风格得以统一。好的母版页设计不光要求编码准确，更要求界面颜色搭配，功能区域布局合理。因此读者在加强编码能力的同时也应该在界面设计方面多下工夫，只有这样才能设计出风格突出、功能强大的网站项目。

3.6.7 练习题

- 1. 调整 VS2008 编辑区，HTML 编码的字号为 10 号字。
- 2. 调整 VS2008 编辑区，C#编码的字号为 12 号字。

3.7 前台用户的登录、注册、个人设置、修改个人资料实现

学习目标

- 掌握数据访问类的制作方法
- 掌握 ConfigurationManager 类获取数据库连接字符串的用法
- 掌握 SqlConnection 类的用法
- 掌握 SqlCommand 类的用法
- 掌握 SqlDataReader 类型方法的编制方法
- 掌握 SqlDataSource 类型方法的编制方法
- 掌握在 web.config 文件中配置数据库连接字符串的方法
- 建议学时：16 学时

3.7.1 任务名称：前台用户的登录、注册、个人设置、修改个人资料实现

3.7.2 任务描述

本任务中初步涉及数据访问层的设计。首先，需要制定业务逻辑层、数据访问层和实体类，根据系统需求，设计项目与数据库访问的接口；然后，设计各类业务所操作的对象实体类，再针对各种操作完成数据访问方法；最后，根据业务逻辑设计业务管理类的方法。

接下来进行表示层的设计，在表示层设计环境中，主要进行网页界面的设计，如各种文本框的摆放布局、按钮显示的文字等。通过各种控件所对应的各类事件处理接口与业务管理类的相关方法相调用，进而完成各个业务操作流程，实现手工业务向电子业务的转化。

主要完成的任务如下：

- 登录功能的实现；
- 注册功能的实现；
- 个人设置；
- 修改个人资料等。

3.7.3 任务分析

该项目中采用的是标准的三层架构，通常的做法是将除表示层以外的其他各层以类库的形式完成，然后在网站项目中加以引用。此种做法在每次对类库项目进行修改通过时需要重新编译类库项目、重新添加引用等，使得代码的调试变得复杂且容易出错。因此为了调试方便，在这里采用文件夹分类的方式，将业务逻辑层和数据访问层及实体类的类用文件夹的方式进行物理上的分类，并使用命名空间进行逻辑上的分类。此种做法也可以作为三层架构设计的一种折中方案，虽然降低了代码的可重用性，但给调试带来了极大的便利。

3.7.4 Model层：用户实体类User类的实现

程序开发步骤如下：

前台用户的登录、注册、个人设置、修改个人资料功能，都是通过对 OL_User 表的操作实现的，因此首先需要创建该表的实体类结构。

在“解决方案资源管理器”中右击项目，选择“添加新项”→“类”，如图 3-22 所示。

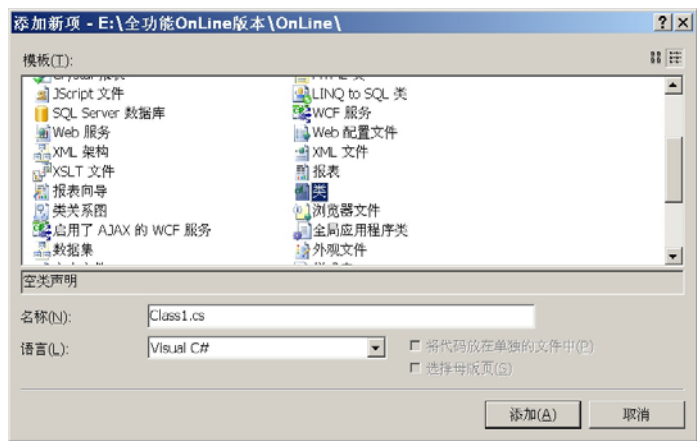


图 3-22 添加类

此时系统提示“是否把该类添加到 App_Code 文件夹下”，单击“是”按钮，目的是让系统自动生成“App_Code”文件夹。

删除刚刚建立的类文件，选中“App_Code”文件夹，在其内部添加一个新的文件夹，命名为“MODEL”。

在该文件夹下添加新项，选择“类”，命名为“User”。

根据 OnLine 数据库中 OL_User 表的字段名称和类型，编写与该表对应的实体类定义代码，代码如下。

```
用户实体类：User.cs
public class User
{
    public User(){}
    //带参数的构造方法
    public User(string name, string online){
        this.name = name;
        this.online = online;
    }
    //用户名称
    private string name;
    public string Name{
        get { return name; }
        set { name = value; }
    }
    //用户密码
    private string password;
    public string Password{
        get { return password; }
        set { password = value; }
    }
}
```

```

        //在线标识
private string online;
public string Online{
    get { return online; }
    set { online = value; }
}
//用户类型： 客服或用户
private string isAdmin;
public string IsAdmin{
    get { return isAdmin; }
    set { isAdmin = value; }
}
}

```

在“App_Code”文件夹下添加新文件夹，取名“DAL”。

3.7.5 DAL层：数据访问类DataBase类的实现

在“DAL”文件夹下添加类，命名为“DataBase”，为数据访问层通用代码类。

1. 数据访问类DataBase.cs

程序代码如下：

```

public class DataBase: IDisposable{
    protected SqlConnection Connection;
    private String ConnectionString;
    public DataBase(){
        //获取 web.config 中保存的名为 conn 的数据库连接字符串
        ConnectionString = ConfigurationManager.AppSettings["conn"];
    }
    //当连接对象资源被系统回收时，调用关闭连接方法
    ~DataBase(){
        try{
            if (Connection != null)
                Connection.Close();
        }
        catch (Exception e){
        }
        try{
            Dispose();
        }
        catch { }
    }
    //打开数据库连接
    protected void Open(){
        if (Connection == null){
            try{
                Connection = new SqlConnection(ConnectionString);
            }
            catch (Exception e){ }
        }
    }
}

```



```

    }
    If (Connection.State.Equals(ConnectionState.Closed))
    {
        try
        {
            Connection.Open();
        }
        catch (Exception e){}
    }
}
//关闭数据库连接
public void Close(){
    try{
        if (Connection != null)
            Connection.Close();
    }
    catch (Exception e){}
}
//释放连接对象 Connection 所占用的系统资源
public void Dispose()
{
    //确保连接被关闭
    try{
        if (Connection != null){
            Connection.Dispose();
            Connection = null;
        }
    }
    catch (Exception e){}
}
//返回 DataReader 数据集
public SqlDataReader GetDataReader(String SqlString){
    Open();
    try{
        SqlCommand cmd = new SqlCommand(SqlString, Connection);
        return cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
    }
    catch (Exception e){
        return null;
    }
}
//执行方法参数 sqlCmdUpdate 代表的 SQL 语句，更新数据库数据
//方法返回值类型为 bool
public bool Update(string sqlCmdUpdate){
    Open();
    try{
        SqlCommand cmd = new SqlCommand(sqlCmdUpdate, Connection);

```

```

        cmd.ExecuteNonQuery();
        return true;
    }
    catch (Exception e){
        return false;
    }
}
//执行方法参数 sqlCmdSelect 代表的 SQL 语句
//方法返回值类型为 SqlDataReader
public SqlDataReader getSqlDataSource(string sqlCmdSelect){
    SqlDataReader sds=new SqlDataReader(Connection, sqlCmdSelect);
    return sds;
}
//执行方法参数 sqlCmdSelect 代表的 SQL 语句，对数据库进行数据删除操作
//方法返回值类型为 bool
public bool Insert(string sqlCmdInsert){
    Open();
    try{
        SqlCommand cmd = new SqlCommand(sqlCmdInsert, Connection);
        cmd.ExecuteNonQuery();
        return true;
    }
    catch (Exception e){
        return false;
    }
}
//执行方法参数 sqlCmdSelect 代表的 SQL 语句，对数据库进行数据删除操作
//方法返回值类型为 bool
public bool Delete(string sqlCmdDelete){
    Open();
    try{
        SqlCommand cmd = new SqlCommand(sqlCmdDelete, Connection);
        cmd.ExecuteNonQuery();
        return true;
    }
    catch (Exception e){
        return false;
    }
}
}

```

在上面的代码中有一处代码读取了位于网站项目根目录下的 web.config 文件，打开该文件，找到<configuration> </configuration>节，在其内部添加以下代码：

```

<appSettings><addkey="conn"value="DataSource=LH\SQLEXPRESS;InitialCatalog=OnLine;Integrated
Security=True"/></appSettings>

```

① DAL（层）文件夹下添加新类，命名为 UserDAL。

② 在 DAL 层 UserDAL 类中添加登录方法 Login。

③ 在 DAL 文件夹下添加类 UserDAL。

在该类中编写用户登录方法 `public bool Login(string name, string password)`。

程序代码如下：

```
public bool Login(string name, string password){
    DataBase db = new DataBase();
    string sqlCommandSelect="select*fromOL_Userwhere name='"+name+"'and password='"+password+"' ";
    SqlDataReader sdr = db.GetDataReader(sqlCmdSelect);
    //用户名、密码正确
    if (sdr.Read()){
        //登录成功，修改在线状态
        return this.UpDateOnLine(name, "在线");
    }
    else{
        //登录失败
        return false;
    }
}
```

2. 在DAL层UserDAL类中添加注册方法Reg

Reg 方法为通用注册方法，当需要注册客服人员时，要先行调用该方法，注册成普通会员，然后再通过后台管理系统提升权限到客服类型。

程序代码如下：

```
//普通用户注册
public bool Reg(string name, string password){
    DataBase db = new DataBase();
    string sqlCommandInsert="insertinto L_User(name,password,isAdmin,online)values('"+name+"',"
+password+"','普通用户','离线)";
    return db.Insert(sqlCmdInsert);
}
```

3. 在DAL层UserDAL类中添加修改用户类型为客服的方法UpdateIsAdmin

程序代码如下：

```
//后台调用，修改用户类型为客服
public bool UpdateIsAdmin(string name){
    DataBase db = new DataBase();
    string sqlCommandUpdate = "update OL_User set isadmin='客服' where name='"+name+"' ";
    return db.Update(sqlCmdUpdate);
}
```

3.7.6 BLL层：业务逻辑类的实现

在“App_Code”文件夹下添加新文件夹，命名为“BLL”，在其内部添加新类，命名为“UserBLL”。

添加登录业务方法 Login、注册业务方法 Reg、用户类型更改业务方法 EditUserType。

程序代码如下：

```
public bool Login(string name, string password){
    return userDAL.Login(name, password);
}

public bool Reg(string name, string password){
    //检测是否存在用户名 name
    if (!userDAL.ExistUser(name)){
        //不存在该用户名 name，插入注册新用户 name
        return userDAL.Reg(name, password);
    }
    else{
        //存在重名，注册失败
        return false;
    }
}

public void EditUserType(string name, string type){
    userDAL.EditUserType(name, type);
}
```

值得注意的是，在执行注册操作之前还需要对数据库用户表 OL_User 中已有的用户进行检索，防止出现待注册用户的名称和已有用户同名。这里使用了 DAL 层中的 ExistUser 方法，该方法在 DAL 层 UserDAL.cs 类中的代码如下：

```
/// <summary>
/// 查询指定用户是否存在
/// </summary>
/// <param name="name">用户名</param>
/// <returns>true 存在或者 false 不存在</returns>
public bool ExistUser(string name){
    DataBase db = new DataBase();
    string sqlCmdSelect = "select * from OL_User where name='" + name + "'";
    SqlDataReader sdr = db.GetDataReader(sqlCmdSelect);
    if (sdr.Read()){
        return true;
    }
    else{
        return false;
    }
}
```

在 BLL 层 UserBLL.cs 类中的所有方法几乎都在使用 DAL 层下的 UserDAL 对象，因此还需要在 UserBLL 类的开始部分定义一个类型为 UserDAL 的数据成员。代码如下：

```
private UserDAL userDAL = new UserDAL();
public UserBLL(){}
```

3.7.7 登录页面表示层代码的实现

在网站项目根目录下添加新项“web 窗体”，命名为“Login”。

Login.aspx 的 HTML 代码为:

[illegible]

Login.aspx 在设计视图模式下的效果如图 3-23 所示。

用户名:	<input type="text"/>
密 码:	<input type="password"/>
<input type="button" value="登录"/>	<input type="button" value="注册"/>

图 3-23 登录页面效果图

打开 Login.aspx.cs 文件，在其内部添加登录按钮的执行方法 btnLogin_Click。

程序代码如下:

```
protected void btnLogin_Click(object sender, EventArgs e){
    string name=txtName.Text.Trim();
    string password=txtPassword.Text.Trim();
    if (name == "" || password == ""){
        //信息不完整
        this.Response.Write("<script language=javascript>alert('用户名或密码不能为空! ');</script>");
    }
    else{
        UserBLL userBLL = new UserBLL();
        bool pass = userBLL.Login(name, password);
        bool isAdmin = userBLL.isAdmin(name);
        if (pass){
            //通过验证
        }
    }
}
```

```

        Session["name"] = name;

        if (isAdmin){
            Response.Redirect("adminManager.aspx");
        }
        else{
            Response.Redirect("ChatRoom.aspx");
        }
    }
    else{
        //验证失败
        txtName.Text = "";
        txtPassword.Text = "";
        this.Response.Write("<scriptlanguage='javascript'>alert('用户名或密码错误! ');</script>");
    }
}
}

```

继续添加注册按钮的执行方法 btnReg_Click。

程序代码如下：

```

protected void btnReg_Click(object sender, EventArgs e){
    string name = txtName.Text.Trim();
    string password = txtPassword.Text.Trim();
    if (name == "" || password == ""){
        //信息不完整
        this.Response.Write("<scriptlanguage='javascript'>alert('用户名或密码不能为空! ');</script>");
    }
    else{
        UserBLL userBLL = new UserBLL();
        bool reg = userBLL.Reg(name, password);
        if (reg == true){
            //注册成功，进入聊天室
            this.Response.Write("<scriptlanguage='javascript'>alert('注册成功, 请重新登录! ');</script>");
        }
        else{
            //注册失败
            txtName.Text = "";
            txtPassword.Text = "";
            this.Response.Write("<script language='javascript'>alert('注册失败, 已存在该用户! ');</script>");
        }
    }
}
}

```

3.7.8 相关技能与知识点

本任务初步涉及了数据访问层的设计，以及网站对数据库数据的访问，相继用到了许多数据访问类。

1. appSettings 元素

appSettings 元素存储自定义应用程序配置信息，如文件路径、XML Web services URL 或存储在应用程序.ini 文件中的任何信息。可以使用 ConfigurationSettings 类在代码中访问 appSettings 元素中指定的键/值对。

用户可以使用 file 属性指定一个配置文件，该配置文件提供其他设置或重写 appSettings 元素中指定的设置。可以将 file 属性用于源代码管理组开发方案，例如，当用户需要重写在应用程序配置文件中指定的项目设置时，在 file 属性中指定的配置文件必须将 appSettings 元素（而不是 configuration 元素）作为根节点。

2. SqlConnection类

SqlConnection 对象表示与 SQL Server 数据源的一个唯一的会话。对于客户端/服务器数据库系统，它等效于到服务器的网络连接。SqlConnection 与 SqlDataAdapter、SqlCommand 一起使用，可以在连接 Microsoft SQL Server 数据库时提高性能。对于所有第三方 SQL 服务器产品及其他支持 OLE DB 的数据源，应使用 OleDbConnection。

若要确保连接始终关闭，可在 using 块内部打开连接，这样可确保在代码退出代码块时自动关闭连接。

3. SqlCommand成员

当创建 SqlCommand 的实例时，读/写属性将被设置为其初始值。

4. SqlDataSource成员

SqlDataSource 数据源控件用于表示绑定到数据绑定控件 SQL 关系数据库中的数据。将 SqlDataSource 控件与数据绑定控件一起使用，可以从关系数据库中检索数据，还可以在网页上显示、编辑和排序数据，而不必编写代码或只需编写少量代码。

5. SqlDataReader类

若要创建 SqlDataReader，必须调用 SqlCommand 对象的 ExecuteReader 方法，而不能直接使用构造函数。

在使用 SqlDataReader 时，关联的 SqlConnection 正忙于为 SqlDataReader 服务，对 SqlConnection 无法执行任何其他操作，只能将其关闭。除非调用 SqlDataReader 的 Close 方法，否则会一直处于此状态。例如，在调用 Close 之前，无法检索输出参数。

当 SqlDataReader 关闭后，只能调用 IsClosed 和 RecordsAffected 属性。尽管当 SqlDataReader 存在时可以访问 RecordsAffected 属性，但是应始终在返回 RecordsAffected 的值之前调用 Close，以保证返回精确的值。

3.7.9 任务小结

ASP.NET 之所以可以快速创建各类网站项目，很大程度上得益于其庞大的类库，通过调用相关类库，可以方便地进行二次开发，使程序员把更多的精力集中在业务逻辑上，而非重复的编写代码上。因此，只有熟练地运用各类常用的系统类库，才能快速开发出健壮、高效的程序。本节所涉及的各种类在相当多的场合都有应用，读者应重点加以学习，进而掌握技

术要领与精华所在。最简单也最实用的方法就是利用好 MSDN 这套工具，其内部的很多代码实例非常具有代表性和权威性。

3.7.10 练习题

在不对页面布局产生影响的前提下，编写用于弹出提示对话框的代码：

```
this.Response.Write(_____);
```

3.8 前台（在线/离线）消息发送、浏览、获取客服列表实现

学习目标

- 掌握 ASP.NET AJAX 技术实现的主要控件 `asp:ScriptManager` 的用法
- 掌握 ASP.NET AJAX 技术实现的主要控件 `asp:Timer` 的用法
- 掌握 ASP.NET AJAX 技术实现的主要控件 `asp:UpdatePanel` 的用法
- 建议学时：16 学时

3.8.1 任务名称：前台（在线/离线）消息发送、浏览、获取客服列表实现

3.8.2 任务描述

用户对在线/离线客服发送消息，当所选择客服在线时，在用户操作的界面上应有所提示，并可以直接对其发出问题，而当所选择的客服处于离线状态时，用户对离线客服发送消息时，系统应给出相应的提示，表示该消息为针对离线客服的离线留言，以体现在线客服系统的功能强大之处。系统应能自动实现聊天信息的刷新，自动获取客服列表，即在线和离线的客服。

系统前台部分主要为用户提供以下几个功能：

- 当用户登录成功后，系统自动显示出当前在线和离线的所有客服名称；
- 自动显示该用户曾经发出的消息内容；
- 自动显示客服人员曾经对该用户发出的消息内容；
- 用户在发出消息之前，需要先选择一个客服人员（在线与离线均可）。

3.8.3 任务分析

本任务的主要难点在于如何实现聊天信息的自动刷新，而且需要强调的是对于整个页面而言只有聊天信息部分及时刷新，其他部分不能跟随刷新，否则将对网络产生比较大的数据传输压力。随着 ASP.NET 功能的不断完善，ASP.NET AJAX 控件在处理页面的异步刷新方面发挥出巨大的作用。

使用 ASP.NET 中的 AJAX 功能可快速创建包含具有响应能力且熟悉的用户界面（UI）元素的网页，以提供丰富的用户体验。AJAX 功能包括客户端脚本库，这些库将跨浏览器的 ECMAScript（JavaScript）和动态的 HTML（DHTML）技术结合在一起，并与基于 ASP.NET 服务器的开发平台集成。通过使用 AJAX 功能，可以改进用户体验并提高 Web 应用程序的效率。

1. 使用ASP.NET AJAX功能的原因

使用 ASP.NET 中的 AJAX 功能，可以生成丰富的 Web 应用程序。与完全基于服务器的 Web 应用程序相比，这些应用程序具有很多优点。支持 AJAX 的应用程序可以提供：

- ① 增强的效率，因为网页的大部分处理工作是在浏览器中执行的；
- ② 熟悉的 UI 元素，如进度指示器、工具提示和弹出窗口；
- ③ 部分页更新，只刷新已发生更改的网页部分；
- ④ 客户端与用于 Forms 身份验证的 ASP.NET 应用程序服务、角色和用户配置文件的集成；
- ⑤ 自动生成的代理类，可简化从客户端脚本调用 Web 服务方法的过程；
- ⑥ 一个框架，可使用户自定义服务器控件以包含客户端功能；
- ⑦ 对大部分流行和常用浏览器的支持，其中包括 Microsoft Internet Explorer、Mozilla Firefox 和 Apple Safari。

2. AJAX控件工具箱

ASP.NET AJAX 控件工具箱是示例和组件的集合，这些示例和组件演示一些可使用 ASP.NET AJAX 控件和扩展程序创建的体验。控件工具箱提供一些示例和功能强大的 SDK，这使创建和重用自定义控件及扩展程序变得很简单。可以从 ASP.NET AJAX 网站下载 ASP.NET AJAX 控件工具包。

消息处理的相关功能涉及读取数据库中 UL_Message 表的信息，因此需要创建 Message 实体类。

3.8.4 Model层：消息实体类Message类的实现

程序开发步骤如下：

在 Model 层添加新类，命名为 Message，具体代码如下。

```
消息实体类：Message.cs
public class Message{
    public Message(){ }
    //消息 ID 编号
    private int id;
    public int Id{
        get { return id; }
        set { id = value; }
    }
    //消息发出者
    private string fromUser;
    public string FromUser{
        get { return fromUser; }
        set { fromUser = value; }
    }
    //消息
    private string msg;
    public string Msg{
        get { return msg; }
        set { msg = value; }
```

```

    }
    //消息接收者
    private string toUser;
    public string ToUser{
        get { return toUser; }
        set { toUser = value; }
    }
    //消息发出时间
    private string dateTime;
    public string DateTime{
        get { return dateTime; }
        set { dateTime = value; }
    }
    //消息实体类的构造方法
    public Message(string fromUser, string msg, string toUser){
        this.fromUser = fromUser;
        this.msg = msg;
        this.toUser = toUser;
    }
    public Message(string fromUser, string msg, string toUser, string online){
        this.fromUser = fromUser;
        this.msg = msg;
        this.toUser = toUser;
        this.online = online;
    }
    //消息接收者状态： 在线或离线
    private string online;
    public string Online{
        get { return online; }
        set { online = value; }
    }
}

```

3.8.5 DAL层：数据访问类MessageDAL类的实现

在 DAL 文件夹添加新类，命名为 MessageDAL，添加获取与自己相关消息集合的方法。

```

//获取对自己的聊天信息
public IList<Message> getChatMessages(string name){
    IList<Message> ChatMessages=new List<Message>();
    //获取针对指定参数 name 的聊天信息
    //即消息发出者为 name 和消息接收者为 name 的所有消息，按时间降序排列
    string sqlCmdSelect="SELECT [fromUser],[msg],[toUser],[dateTime]FROM[OL_Message]
where toUser='"+ name+"'or fromuser='"+name+"' order bydatetimedesc";
    SqlDataReader sdr=db.GetDataReader(sqlCmdSelect);
    //将所有消息记录封装到 IList<Message>集合类对象中

```

```

while (sdr.Read()){
    Message msg = new Message();
    msg.FromUser = sdr.GetString(0);
    msg.Msg = sdr.GetString(1);
    msg.ToUser = sdr.GetString(2);
    msg.DateTime = sdr.GetDateTime(3).ToString();
    ChatMessages.Add(msg);
}
//返回所有符合查询条件的聊天记录
return ChatMessages;
}

```

在 DAL 层 UserDAL 类中添加获取客服列表的方法 `getUserList`。

```

//获取客服列表
public IList<User> getUserList(string userType){
    IList<User> userList = new List<User>();
    string sqlCommandSelect;
    if (userType == "客服"){
        sqlCommandSelect = "select name, online from OL_User where isadmin='客服' ";
    }
    else{
        sqlCommandSelect = "select name, online from OL_User where isadmin='普通用户' ";
    }
    DataBase db = new DataBase();
    SqlDataReader sdr = db.GetDataReader(sqlCommandSelect);
    while (sdr.Read()){
        User user = new User(sdr.GetString(0), sdr.GetString(1));
        userList.Add(user);
    }
    return userList;
}

```

在 DAL 层 UserDAL 类中添加用户是否在线的方法 `OnLine`。

```

//用户是否在线
public bool OnLine(string userName){
    DataBase db = new DataBase();
    string sqlCommandSelect = "select online from OL_User where name='" + userName + "' ";
    SqlDataReader sdr=db.GetDataReader(sqlCommandSelect);
    if (sdr.Read()){
        if (sdr.GetString(0) == "在线"){
            return true;
        }
        else{
            return false;
        }
    }
}

```

```

        else{
            return false;
        }
    }
}

```

在 DAL 层 MessageDAL 类中添加发送离线消息的方法 leaveLineSend。

```

/// <summary>
/// 发送离线消息
/// </summary>
/// <param name="msg"></param>
/// <returns></returns>
public bool leaveLineSend(Message msg){
    string sqlCmdInsert="insertintoOL_Message(fromUser,msg,toUser,online)values('"+ msg.FromUser
+ "', '"+ msg.Msg + "', '"+ msg.ToUser + "','离线')";
    return db.Insert(sqlCmdInsert);
}

```

在 DAL 层 MessageDAL 类中添加发送在线信息到数据库的方法 onLineSend。

```

/// <summary>
/// 发送在线信息到数据库
/// </summary>
/// <param name="msg"></param>
/// <returns></returns>
public bool onLineSend(Message msg){
    string sqlCmdInsert="insertintoOL_Message(fromUser,msg,toUser,online)values ('"+msg.FromUser
+ "', '"+msg.Msg+"','"+msg.ToUser+"','在线')";
    return db.Insert(sqlCmdInsert);
}

```

在 DAL 层 UserDAL 类中添加退出系统的方法 ExitLogin。

```

//退出系统
public bool ExitLogin(string name){
    DataBase db = new DataBase();
    string sqlCmdUpdate = "update OL_User set online='离线' where name='"+ name + "' ";
    return db.Update(sqlCmdUpdate);
}

```

3.8.6 BLL层：业务逻辑类的实现

在 BLL 文件夹添加新类 MessageBLL，定义 MessageDAL 类型的对象，再添加与上面各个方法对应的业务管理方法。

```

private MessageDAL messageDAL = new MessageDAL();
public IList<Message> getChatMessages(string name){
    return messageDAL.getChatMessages(name);
}

```

```
//发送离线留言
public bool LeavaLineSend(Message msg){
    return messageDAL.leaveLineSend(msg);
}

//发送在线消息
public bool Send(Message msg){
    return messageDAL.onLineSend(msg);
}
```

在 BLL 文件夹的 UserBLL 类中添加与上面各个方法对应的业务管理方法（如果该方法已存在，则表示该功能需要调用该方法）。

```

        public IList<User> getUserList(string userType){
            return userDAL.getUserList(userType);
        }

        public bool OnLine(string userName){
            return userDAL.OnLine(userName);
        }

        public bool ExitLogin(string name){
            return userDAL.ExitLogin(name);
        }
    }
}

```

3.8.7 消息发送、消息浏览表示层代码的实现

在网站根目录下添加“web 窗体”，命名为“ChatRoom”。在 ChatRoom.aspx 的 HTML 源代码模式下添加如下代码：

```
<body>
    <form id="form1" runat="server">
    < asp:ScriptManager ID="ScriptManager1" runat="server"/>
    <asp:Timer ID="Timer1" runat="server" Interval="1000" OnTick="Timer1_Tick"></asp:Timer>
    <asp:Timer ID="Timer2" runat="server" Interval="1000" OnTick="Timer2_Tick"></asp:Timer>
    <div class="DivClass"><table class="TableClass">
        <tr><td class="tr1_td1_td2" colspan="2"></td></tr>
        <tr><td class="tr2_td1"><asp:UpdatePanel ID="UpdatePanel1" runat="server" >
            <ContentTemplate><asp:Repeater ID="Repeater1" runat="server"><HeaderTemplate>
                <div style="text-align:left; overflow-y:scroll; height: 300px;"></HeaderTemplate>
                <ItemTemplate><asp:LabelID="lblDateTime"runat="server"Text=""ForeColor="#3333FF">
<%# Eval("DateTime")%></asp:Label>&nbspsp;&nbspsp;
                    <asp:LabelID="lblFromUser"runat="server"Text=""ForeColor="#339966"><%# Eval
("fromUser") %></asp:Label>对
                    <asp:LabelID="lblToUser"runat="server"Text=""ForeColor="Red"><%# Eval("ToUser")
%></asp:Label>说: <br />
                    <asp:Label ID="lblMsg" runat="server" Text="" ForeColor="#993300">&nbspsp;&nbspsp;
<%# Eval("Msg")%></asp:Label>
                    <br /></ItemTemplate><FooterTemplate></div></FooterTemplate></asp:Repeater>
                </ContentTemplate>
            </td></tr>
    </table></div>
    </form>
</body>
```

```

        <Triggers><asp:AsyncPostBackTriggerControlID="Timer1"EventName="Tick"/></Triggers>
    </asp:UpdatePanel></td><tdclass="tr2_td2"><asp:UpdatePanelID="UpdatePanel2" runat="server">
        <ContentTemplate><asp:Repeater ID="rpUserList" runat="server">
            <HeaderTemplate> <div style="text-align:left; overflow-y:scroll; height: 300px;">
            </HeaderTemplate>
            <ItemTemplate>
                <imgalt=""src="img/kefu.jpg"/><ahref="ChatRoom.aspx?adminName=<%#Eval
("name")%>"/><%#Eval("name")%>[<%#Eval("online")%>]<br/></ItemTemplate>
<FooterTemplate></div></FooterTemplate></asp:Repeater></ContentTemplate>
        <Triggers><asp:AsyncPostBackTriggerControlID="Timer2"EventName="Tick" /></Triggers>
        </asp:UpdatePanel>
    </td></tr>
    <tr><tdclass="tr3_td1">我对<asp:LabelID="lblToMsgUserName"runat="server"Text="" >
</asp:Label>说:</td>
        <td class="tr3_td2"><asp:Button ID="btnStop" runat="server" Width="75px" Text="停止刷新"
            onclick="btnStop_Click" /><asp:Button ID="btnExitLogin" runat="server"
Text="退出系统" onclick="btnExitLogin_Click" /></td></tr>
        <tr><tdclass="tr4_td1"><asp:TextBoxID="txtMsg" runat="server" Width="400px"Rows="2"
TextMode="MultiLine"></asp:TextBox></td>
            <tdclass="tr4_td2"><asp:ButtonID="btnSend"Text="发送"runat="server" Width="75px"
            onclick="btnSend_Click" /></td></tr></table></div></form>

</body>

```

打开 ChatRoom.aspx.cs 文件，添加如下代码：

```

public partial class ChatRoom : System.Web.UI.Page{
    private readonly MessageBLL messageBLL = new MessageBLL();
    private readonly UserBLL userBLL = new UserBLL();
    protected void Page_Load(object sender, EventArgs e){
        if (!IsPostBack){
            if (Session.Count == 0){
                Response.Redirect("login.aspx");
            }
            string adminName = Request.QueryString["adminName"];
            lblToMsgUserName.Text = adminName;
        }
    }
    protected void Timer1_Tick(object sender, EventArgs e){
        string name = Session["name"].ToString();
        Repeater1.DataSource = messageBLL.getChatMessages(name);
        Repeater1.DataBind();
    }
    protected void Timer2_Tick(object sender, EventArgs e){
        rpUserList.DataSource = userBLL.getUserList("客服");
        rpUserList.DataBind();
    }
    protected void btnStop_Click(object sender, EventArgs e){

```

```

        if (btnStop.Text == "停止刷新"){
            btnStop.Text = "刷新";
            Timer1.Enabled = false;
            Timer2.Enabled = false;
        }
        else{
            btnStop.Text = "停止刷新";
            Timer1.Enabled = true;
            Timer2.Enabled = false;
        }
    }

    protected void btnSend_Click(object sender, EventArgs e){
        string strMsg = txtMsg.Text.Trim();
        if (strMsg == ""){
            this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert
('不能发送空消息')</script>");
        }
        else{
            if (strMsg.Length > 40){
                txtMsg.Text = "";
                this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert
('不能发送大于 40 个字符的消息')</script>");
            }
            else{
                //必须选择提问客服后才能发送消息
                //没有选择客服
                if (lblToMsgUserName.Text == ""){
                    this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert
('请在右侧选择客服')</script>");
                }
                else{
                    //选择客服后发送消息
                    //保存自身用户名
                    string name = Session["name"].ToString();
                    //客服是否在线
                    bool online=userBLL.OnLine(lblToMsgUserName.Text);
                    Message msg = new Message(name, strMsg, lblToMsgUserName.Text);
                    //客服离线时，先接收离线消息
                    if (!online){
                        //获取该客服的离线留言
                        //stringleaveWord= messageBLL.getLeaveWord(lblToMsgUserName.Text);
                        //当存在离线消息时，弹出离线消息对话框
                        bool leaveWordSend = messageBLL.LeaveLineSend(msg);
                        if (leaveWordSend == true){
                            this.ClientScript.RegisterStartupScript (this.GetType(),"", "<script>
window.alert('您对"+lblToMsgUserName.Text+ "的离线留言发送成功!')</script>");
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
}
else{
    //发送消息给该客服
    messageBLL.Send(msg);
    txtMsg.Text = "";
}
}
}
}
}
protected void btnExitLogin_Click(object sender, EventArgs e){
    string name = Session["name"].ToString();
    Response.Write(userBLL.ExitLogin(name));
}
{
{

```

3.8.8 相关技能与知识

本任务的最终实现主要得益于 ASP.NET AJAX 控件的合理运用，该控件组包含的控件很多，比较常用的主要是 asp:ScriptManager 控件、asp:Timer 控件和 asp:UpdatePanel 控件。

1. ScriptManager控件

ScriptManager控件是ASP.NET中AJAX功能的中心。该控件可管理一个页面上的所有ASP.NET AJAX资源，其中包括将Microsoft AJAX Library脚本下载到浏览器和协调通过使用UpdatePanel 控件启用的部分页面更新。

2. Timer控件

当要执行以下操作时，可使用 Timer 控件：

- ① 定期更新一个或多个 UpdatePanel 控件的内容，而无须刷新整个网页；
- ② 每当 Timer 控件导致回发时运行服务器上的代码；
- ③ 按定义的时间间隔将整个网页同步发布到 Web 服务器上。

Timer 控件是一个服务器控件，它将一个 JavaScript 组件嵌入网页中，当经过 Interval 属性中定义的时间间隔时，该 JavaScript 组件将从浏览器启动回发。用户可以在运行于服务器上的代码中设置 Timer 控件的属性，这些属性将传递到该 JavaScript 组件。

使用 Timer 控件时，必须在网页中包括 ScriptManager 类的实例。

若回发是由 Timer 控件启动的，则 Timer 控件将在服务器上引发 Tick 事件。该页发送到服务器时，可以创建 Tick 事件的事件处理程序来执行一些操作。

设置 Interval 属性可指定回发发生的频率，而设置 Enabled 属性可打开或关闭 Timer。Interval 属性是以 ms 为单位定义的，其默认值为 60 000ms（即 60s）。

3. UpdatePanel控件

UpdatePanel 控件是一个服务器控件，可帮助用户开发具有复杂的客户端行为的网页，使网页与最终用户之间具有更强的交互性。若要编写用于在服务器和客户端之间进行协调的代码及更新网页的指定部分，通常需要深入了解 ECMAScript（JavaScript）。不过，通过使用

UpdatePanel 控件，可以使网页参与到部分页的更新中，而无须编写任何客户端脚本。如果需要，可以添加自定义客户端脚本以增强客户端用户体验。当使用 UpdatePanel 控件时，页行为是独立于浏览器的，并且有可能会减少在客户端和服务器之间传输的数据量。

3.8.9 任务小结

在实现页面的局部刷新过程中，使用了 ASP.NET AJAX 控件组中的 3 个主要控件，简单设置了其相关属性便实现了该功能。而在以往单纯依靠 JavaScript 脚本语言实现页面局部刷新的设计中，需要数倍于此的代码量为基础，可见 ASP.NET AJAX 组件功能之强大。在学习过程中切忌全学、全面掌握，这在当前的初学者中尤为普遍，很多技术都在不断进步、不断更新，只要把眼前需要的学会、掌握就可以解决很多实际问题，许多以前看起来非常先进的技术，随着时间的推移渐渐被其他更为科学、更为进步的技术所替代。正如我们不需要去对每一个控件、每一个属性、每一个方法全面掌握一样。在实际的开发过程中，真正能够长久提供技术支援的，多半是开发技术的技术支持功能，如.NET 平台的 MSDN、Java 平台的 API 等。重点掌握以上 3 个控件的基本设置与应用，便可以满足绝大部分需要采用异步刷新技术的场合与应用。

3.8.10 练习题

设定 UpdatePanel1 控件采用 Timer1 控件的刷新方式，填补下列空缺，使其完整。

```
<form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server"/>
    <asp:Timer ID="Timer1" runat="server" Interval="1000" OnTick="Timer1_Tick"></asp:Timer>
    <asp:Timer ID="Timer2" runat="server" Interval="1000" OnTick="Timer2_Tick"></asp:Timer>
    <div class="DivClass"><table class="TableClass"><tr >
        <td class="tr1_td1_td2" colspan="2"></td></tr>
        <tr ><td class="tr2_td1" ><asp:UpdatePanel ID="UpdatePanel1" runat="server" >
            <ContentTemplate>          </ContentTemplate>
            <Triggers>_____</Triggers>
        </asp:UpdatePanel>
    </td></pre>
```

3.9 后台客服管理

学习目标

- 掌握带参数的 SQL 查询语句字符串定义方法
- 建议学时：8 学时

3.9.1 任务名称：后台客服管理

3.9.2 任务描述

客服人员通过设定待修改用户名称及用户类型，将客服转变为普通用户，或者将普通用户转变为客服人员。

本节主要实现如下功能：

- 提升用户权限到客服；
- 降低客服权限到用户；
- 删除客服。

3.9.3 任务分析

本任务要实现的 3 个功能可以在一个 web 窗体下实现，其中前两个功能可以合并为一个功能模块，后一个独立实现。

3.9.4 Model层：实体类的实现

本节的功能全部依靠 Model 层的 User 实体类实现，其定义代码已经完成，故此不再重复。

3.9.5 DAL层：数据访问类的实现

在 DAL 层 UserDAL 类中添加获取所有用户的方法 `getAllUser`。

```
public IList<User> getAllUser(){
    IList<User> alluser = new List<User>();
    string sqlCmdSelect;
    sqlCmdSelect = "select name from OL_User";
    DataBase db = new DataBase();
    SqlDataReader sdr = db.GetDataReader(sqlCmdSelect);
    while (sdr.Read()){
        User u = new User();
        u.Name = sdr.GetString(0);
        alluser.Add(u);
    }
    return alluser;
}
```

在 DAL 层 UserDAL 类中添加获取所有客服的方法 `getAllAdmin`。

```
public IList<User> getAllAdmin(){
    IList<User> AdminList = new List<User>();
    string sqlCmdSelect;
    sqlCmdSelect = "select name from OL_User where isadmin='客服'";
    DataBase db = new DataBase();
    SqlDataReader sdr = db.GetDataReader(sqlCmdSelect);
    while (sdr.Read()){
        User u = new User();
        u.Name = sdr.GetString(0);
        AdminList.Add(u);
    }
    return AdminList;
}
```

在 DAL 层 UserDAL 类中添加设定用户类型为客服或普通用户的方法 EditUserType。

```
//设定用户类型为客服或普通用户
public void EditUserType(string name, string type){
    DataBase db = new DataBase();
    string sqlCmdUpdate = "update OL_User set isadmin='"+type+"' where name='"+ name + "' ";
    db.Update(sqlCmdUpdate);
}
```

在 DAL 层 UserDAL 类中添加删除指定名称客服的方法 deleteAdmin。

```
//删除指定名称的客服
public void deleteAdmin(string name){
    DataBase db = new DataBase();
    string sqlCmdDelete = "delete from OL_User where name='"+ name + "'";
    db.Delete(sqlCmdDelete);
}
```

3.9.6 BLL层：业务逻辑类的实现

在 BLL 层的 UserBLL 类中添加与上面方法对应的业务管理方法。

```
public IList<User> getAllUser()
{
    return userDAL.getAllUser();
}
public IList<User> getAllAdmin()
{
    return userDAL.getAllAdmin();
}
public void EditUserType(string name, string type)
{
    userDAL.EditUserType(name, type);
}
public void deleteAdmin(string name)
{
    userDAL.deleteAdmin(name);
}
```

3.9.7 客服管理表示层代码的实现

在网站项目根目录下添加“web 窗体”，命名为“adminkfgl”。

在 adminkfgl.aspx 的源代码视图模式下，添加如下 HTML 代码：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<table style="border-color:Green; border-style:solid">
```

```

<tr><td><p>提示:添加新客服,需要先行在登录页面注册新用户!</p>
选择用户:<asp:DropDownList ID="ddlUser" runat="server"></asp:DropDownList><br />
修改角色:<asp:DropDownList ID="ddlEdit" runat="server"></asp:DropDownList><br />
<asp:Button ID="btnEdit" runat="server" Text="修改角色" onclick="btnEdit_Click" /></td>
<td>选择客服:<asp:DropDownList ID="ddlDele" runat="server"></asp:DropDownList><br />
<asp:ButtonID="btnDelete"runat="server"Text="删除选中客服"onclick="btnDelete_Click" />
</td></tr></table></asp:Content>

```

打开 adminkfgl.aspx.cs 文件, 添加如下代码:

```

public partial class adminkfgl : System.Web.UI.Page{
    UserBLL userBLL = new UserBLL();
    protected void Page_Load(object sender, EventArgs e){
        if (!IsPostBack){
            this.myDataBind();
        }
    }
    protected void myDataBind(){
        ddlUser.Items.Clear();
        ddlEdit.Items.Clear();
        ddlDele.Items.Clear();
        IList<User> users = userBLL.getAllUser();
        foreach (User u in users){
            ddlUser.Items.Add(u.Name);
        }
        ddlEdit.Items.Add("普通用户");
        ddlEdit.Items.Add("客服");
        IList<User> alladmin = userBLL.getAllAdmin();
        foreach (User u in alladmin){
            ddlDele.Items.Add(u.Name);
        }
    }
    protected void btnEdit_Click(object sender, EventArgs e){
        string name = ddlUser.SelectedValue.ToString();
        string type = ddlEdit.SelectedValue.ToString();
        userBLL.EditUserType(name, type);
        this.myDataBind();
    }
    protected void btnDelete_Click(object sender, EventArgs e){
        string name=ddlDele.SelectedValue.ToString();
        userBLL.deleteAdmin(name);
        this.myDataBind();
    }
}

```

3.9.8 任务小结

本任务涉及的技术较为简单，且多数技术在前面章节都已介绍。在本项目中，所有关于数据库查询的 SQL 语句的代码都是在程序内部实现的，当需要编写带有参数作为查询条件的 SQL 语句时要注意书写规范，特别是查询字段为 varchar 或 nvarchar 类型时的 SQL 语句的书写方式。

3.9.9 练习题

写出 User 表中 name 值为“tt”的 SQL 查询语句。

```
Public void getUserByName(string tt)
{
    ...
    String sqlCmd=_____;
```

3.10 后台消息管理实现

学习目标

- 掌握 GridView 删除按钮功能的实现方法
- 建议学时：8 学时

3.10.1 任务名称：后台消息管理实现

3.10.2 任务描述

本节的主要功能为：浏览在线客服系统中所有用户与客服人员之间的交流信息，并根据实际需要删除相关记录。

3.10.3 任务分析

本节任务为在浏览用户与客服人员之间交流信息的同时可以进行删除相关记录的操作。ASP.NET 的数据显示控件中，GridView 控件具有浏览的同时支持对数据操作的能力，因此，主要利用 GridView 控件生成删除按钮的能力，在其删除按钮的 RowDeleting 事件中完成对选定记录删除的操作。

3.10.4 Model层：实体类的实现

本功能主要使用 Model 层的 Message 实体类作为数据传值对象，而该实体类定义部分前面章节已经给出，故不再重复。

3.10.5 DAL层：数据访问类的实现

在 DAL 层的 MessageDAL 类中添加用来获取用户与客服之间所有交流信息的方法 getGridViewDataSource。

```

public IList<Message> getGridViewDataSource(){
    IList<Message> ChatMessages = new List<Message>();
    stringsqlCmdSelect="SELECT[fromUser],[msg],[toUser],[dateTime],[id] FROM[OL_Message]
order by datetime desc";
    SqlDataReader sdr = db.GetDataReader(sqlCmdSelect);
    while (sdr.Read()){
        Message msg = new Message();
        msg.FromUser = sdr.GetString(0);
        msg.Msg = sdr.GetString(1);
        msg.ToUser = sdr.GetString(2);
        msg.DateTime = sdr.GetDateTime(3).ToString();
        msg.Id = sdr.GetInt32(4);
        ChatMessages.Add(msg);
    }
    return ChatMessages;
}

```

在 DAL 层的 MessageDAL 类中定义根据消息 ID 删除该条消息的方法 Delete。

```

public bool Delete(string id){
    return messageDAL.Delete(id);
}

```

3.10.6 BLL层：业务逻辑类的实现

在 BLL 层的 MessageBLL 类中定义与以上操作相对应的业务管理方法。

```

public IList<Message> getGridViewDataSource(){
    return messageDAL.getGridViewDataSource();
}
public bool Delete(string id){
    return messageDAL.Delete(id);
}

```

3.10.7 消息管理表示层代码的实现

在网站根目录下添加“web 窗体”，命名为“adminxxgl”，并在该文件中添加如下代码：

```

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <asp:GridView ID="GridView1" runat="server" AllowPaging="True"
        OnPageIndexChanging="GridView1_PageIndexChanging" PageSize="5"
        AutoGenerateDeleteButton="True" OnRowDeleting="GridView1_RowDeleting"></asp:GridView>
</asp:Content>

```

打开 adminxxgl.aspx.cs 文件，添加如下代码：

```

public partial class adminxxgl : System.Web.UI.Page{

```

```

protected readonly MessageBLL messageBLL = new MessageBLL();
protected void Page_Load(object sender, EventArgs e){
    if (!IsPostBack){
        this.myDataBind();
    }
}
protected void myDataBind(){
    GridView1.DataSource = messageBLL.getGridViewDataSource();
    GridView1.DataKeyNames = new string[] { "id" };
    GridView1.DataBind();
}
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e){
    string id = GridView1.DataKeys[e.RowIndex].Value.ToString();
    bool delete = messageBLL.Delete(id);
    if (delete == true){
        this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert
('删除成功!')</script>");
    }
    else{
        this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert
('删除失败!')</script>");
    }
    this.myDataBind();
}
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e){
    GridView1.PageIndex = e.NewPageIndex;
    this.myDataBind();
}
}

```

3.10.8 任务小结

熟练掌握 GridView 各项事件的有效处理，可以高效地完成各类数据操作任务。

1. GridView.RowDeleting事件

单击某一行的“删除”按钮后，在 GridView控件删除该行之前，将引发RowDeleting事件。这使用户可以提供一个这样的事件处理方法，即每次发生此事件时就执行一个自定义例程（如取消删除操作）。

GridViewDeleteEventArgs对象将处理方法传递给事件，以便用户可以确定当前行的索引，还可以指示是否应取消删除操作。若要取消删除操作，可将 GridViewDeleteEventArgs对象的 Cancel属性设置为true。如果有必要，还可以在将值传递给数据源之前操作 Keys和 Values集合。

2. GridView.RowDeleted事件

单击某一行的“删除”按钮后，在 GridView控件删除该行之后，将引发RowDeleted事件。这使用户可以提供一个这样的事件处理方法，即每次发生此事件时就执行一个自定义例程（如检查删除操作的结果）。

GridViewDeletedEventArgs 对象将处理方法传递给事件，以使用户可以确定受影响的行数及可能已经发生的任何异常。还可以通过设置 GridViewDeletedEventArgs 对象的 ExceptionHandled 属性来指示是否在事件处理方法中处理了该异常。

3.10.9 练习题

完善以下代码，使得选中单击 GridView 控件中某行所在的删除按钮即可删除该条记录。

```
protected bool GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e){
    string id = GridView1.DataKeys[_____].Value.ToString();
    bool delete = messageBLL.Delete(id);
    if (delete == true){
        Return true;
    }
    else{
        Return false;
    }
}
```

3.11 客服消息发送管理实现

学习目标

- 掌握 readonly 关键字的用法
- 建议学时：8 学时

3.11.1 任务名称：客服消息发送管理实现

3.11.2 任务描述

构建客服工作台的操作界面，该界面需要提供客服与用户实时的交谈信息，客服只能看到对自己发送过消息的用户名单，而不能看到其他的用户名单，只能进行一对一的在线交流服务。

该界面主要包含 3 个功能：

- 显示对该客服提问的所有消息；
- 显示对该客服提问的所有用户名称；
- 选择某一个用户名称对其进行在线交流服务。

3.11.3 任务分析

本节涉及内容与前面章节内容略有重复，区别之处在于，前面章节中的用户可以从在线交流界面中看到在线与离线状态下的客服情况，而本节里的客服人员在工作台界面中只能查看曾经对自己发送过消息的用户情况，无法查看其他用户，这就使得在编写相应的 SQL 语句时要区别对待。


```

        </ContentTemplate>
        <Triggers><asp:AsyncPostBackTriggerControlID="Timer2" EventName="Tick" />
        </Triggers>
    </asp:UpdatePanel></td></tr>
    <tr><td class="tr3_td1">我对<asp:Label ID="lblToMsgUserName" runat="server"Text="" >
</asp:Label>说:</td><td class="tr3_td2">
        <asp:Button ID="btnStop" runat="server" Width="75px" Text="停止刷新"
            onclick="btnStop_Click" />
        <asp:ButtonID="btnExit"runat="server"Text="退出"onclick="btnExit_Click" /></td></tr>
    <td class="tr4_td1"><asp:TextBoxID="txtMsg"runat="server"Width="400px"Rows="2"
        TextMode="MultiLine"></asp:TextBox></td>
        <tdclass="tr4_td2"><asp:ButtonID="btnSend"Text="发送"runat="server" Width="75px"
            onclick="btnSend_Click" /></td></tr></table></div>

```

打开 adminingzt.aspx.cs 文件，添加以下代码：

```

public partial class adminingzt : System.Web.UI.Page{
    private readonly MessageBLL messageBLL = new MessageBLL();
    private readonly UserBLL userBLL = new UserBLL();
    protected void Page_Load(object sender, EventArgs e){
        if (!IsPostBack){
            if (Session.Count == 0){
                Response.Redirect("login.aspx");
            }
            //获取右侧消息接收者，并对消息接收人标签赋值
            string userName = Request.QueryString["userName"];
            lblToMsgUserName.Text = userName;
        }
    }
    protected void Timer1_Tick(object sender, EventArgs e){
        //获取登录页用户名称
        string name = Session["name"].ToString();
        //获取所有自己发出或对自己发出的消息
        Repeater1.DataSource = messageBLL.getChatMessages(name);
        Repeater1.DataBind();
    }
    protected void Timer2_Tick(object sender, EventArgs e){
        rpUserList.DataSource = userBLL.getUserList("普通用户");
        rpUserList.DataBind();
    }
    protected void btnStop_Click(object sender, EventArgs e){
        if (btnStop.Text == "停止刷新"){
            btnStop.Text = "刷新";
            Timer1.Enabled = false;
            Timer2.Enabled = false;
        }
    }
    else{

```

```

        btnStop.Text = "停止刷新";
        Timer1.Enabled = true;
        Timer2.Enabled = false;
    }
}

protected void btnSend_Click(object sender, EventArgs e){
    string strMsg = txtMsg.Text.Trim();
    if (strMsg == ""){
        this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert
('不能发送空消息')</script>");
    }
    else{
        if (strMsg.Length > 40){
            txtMsg.Text = "";
            this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert
('不能发送大于 40 个字符的消息')</script>");
        }
        else{
            //必须选择提问用户后才能发送消息
            //没有选择用户
            if (lblToMsgUserName.Text == ""){
                this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>window.alert
('请在右侧选择用户名称')</script>");
            }
            else
            //选择用户后发送消息{
                //保存自身用户名
                string name = Session["name"].ToString();
                //用户是否在线
                bool online = userBLL.OnLine(lblToMsgUserName.Text);
                Message msg = new Message(name, strMsg, lblToMsgUserName.Text);
                //用户离线时，先接收离线消息
                if (!online){
                    //获取该用户的离线留言
                    //当存在离线消息时，弹出离线消息对话框
                    bool leaveWordSend = messageBLL.LeaveLineSend(msg);
                    if (leaveWordSend == true){
                        this.ClientScript.RegisterStartupScript(this.GetType(), "", "<script>
window.alert('您对 " + lblToMsgUserName.Text + " 的离线留言发送成功!')</script>");
                    }else{
                        //发送消息给该用户
                        messageBLL.Send(msg);
                        txtMsg.Text = "";
                    }
                }
            }
        }
    }
}

```

```
    }  
    protected void btnExit_Click(object sender, EventArgs e){  
        string name = Session["name"].ToString();  
        userBLL.ExitLogin(name);  
    }  
}
```

3.11.7 相关技能与知识

当需要在一个类中反复使用同一个类型对象的各个方法时，不妨定义一个用 `readonly` 修饰的对象。`readonly` 关键字是可以在字段上使用的修饰符。当字段声明包括 `readonly` 修饰符时，该声明引入的字段赋值只能作为声明的一部分出现，或者出现在同一类的构造函数中。

3.11.8 任务小结

合理地运用 `session` 对象，可以临时保存用户的相关信息。若需要长久保存，最好使用个性化配置类 `profile`。

`profile` 使用 `provider` 模式存储信息，默认情况下，`user profile` 的内容会保存在 `SQL Server Express` 数据库中，该数据库位于网站的 `App_Data` 目录下。用户也可以自定义其他数据提供者（`data provider`）来存储信息，如完整版 `SQL Server` 中的一个数据库或者一个 `Oracle` 数据库。

`session` 数据仅仅是存储在服务器的内存中，当用户结束此次访问，或关闭浏览器，则服务器会自动清空该片内存，显然存储在其中的 `session` 数据也随即消失。

3.11.9 练习题

1. 被 `readonly` 关键字修饰的类或字段，其值可以在任何地方被重新赋值。这一说法正确吗？为什么？
2. 写出用以获取当前 `session` 类中变量个数的语句。

第 4 章 商业网站流量统计分析系统

4.1 系统设计

学习目标

■ 掌握 GridView 控件的用法

■ 掌握 Web 用户控件的用法

■ 掌握数据绑定语句 Eval 的用法

■ 建议学时：12 学时

4.1.1 任务名称：系统设计

4.1.2 任务描述


以网上购书商业网站为例，在网站运行期间及时了解用户访问网站的栏目、访问时间、停留时间等数据，可以有效管理网站中的数据，进而规划网站所展示的信息。通过按时间段检索流量表中的数据，将查询结果以表格的形式显示在页面上，同时利用图形化的方式将具体访问流量描述出来，给人以直观的视觉认识，便于网站管理者更为快捷、轻松地了解网站的运行状态及用户访问网站的情况。

商业网站流量统计分析模块的主要功能有：

- 根据指定年份查看网站流量统计；
- 根据指定时间段查看栏目流量统计；
- 根据指定时间段查看 IP 访问统计。

4.1.3 任务分析

（1）按时间段检索表中的数据显示在页面上

利用 ASP.NET 3.5 架构下的 asp:GridView 控件和 asp:SqlDataSource 控件的智能标记功能 ，快速完成数据检索及显示的设置。

（2）根据访问量的不同区别设计对应图例的样式

通过使用数据绑定语法 Eval 调用 asp:SqlDataSource 控件检索到的数据字段，将该值作为图例的宽度，即可根据访问量的多少动态显示图例效果。

4.1.4 任务准备

首先建立访问流量信息数据库。

对于一个通用的商业网站流量统计分析系统而言，网站的管理者通常对访问者的以下信息感兴趣，如访问时间、访问者 IP 地址、访问页面名称、页面停留时间等。而这些信息都是

在用户离开该网站的某个页面时自动获取并保存起来的，通常情况下会使用数据库对这些数据进行保存。根据以上数据特点，下面首先建立访问流量数据库。

① 启动 SQL server 2005, 右键单击左侧栏中的数据库, 选择“新建数据库”命令, 如图 4-1 所示。



图 4-1 新建数据库

② 命名数据库为“traffic”，如图 4-2 所示。

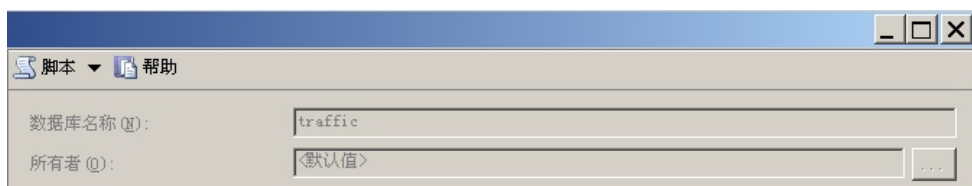


图 4-2 命名数据库

③ 用鼠标右键单击“traffic 数据库”下面的“表”，选择“新建表”命令，如图 4-3 所示。

④ 设定字段类型及名称,如图 4-4 所示。



图 4-3 新建表


LH\SQLEXPRESS.traffic - dbo.Func			摘要
	列名	数据类型	允许空
▶	FuncName	varchar(50)	<input checked="" type="checkbox"/>
	Ip	varchar(50)	<input checked="" type="checkbox"/>
	LoadTime	datetime	<input checked="" type="checkbox"/>
	Minute	int	<input checked="" type="checkbox"/>
	Second	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

图 4-4 设定字段类型及名称

⑤ 其中“LoadTime”字段需要定义为具有默认值属性，“默认值或绑定”设为“getdate()”，如图 4-5 所示。



图 4-5 设定默认值

⑥ 单击快捷方式按钮，指定表名为“Func”。至此数据库建立完毕，接下来进入网站编码部分。

4.1.5 创建商业网站流量分析系统网站母版页

程序开发步骤如下：

① 启动“Visual Studio Team System 2008”，单击菜单栏上的“新建”→“网站”，打开“新建网站”对话框，如图 4-6 所示。

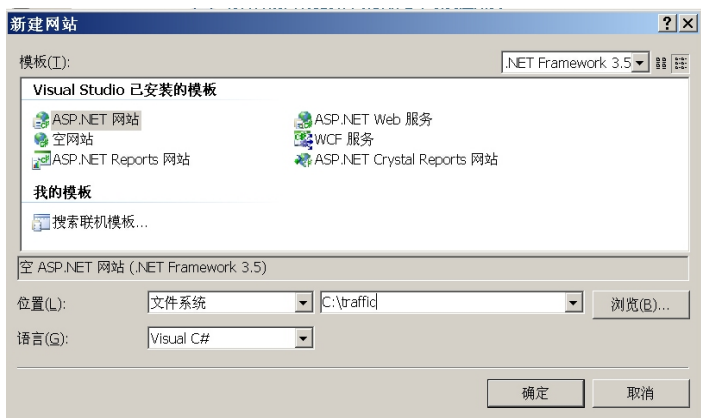


图 4-6 新建网站



在对话框中选择 ASP.NET 网站，指定网站位置、语言及项目名称，如图 4-7 所示。



图 4-7 指定网站位置、语言等信息

② 单击快捷按钮，在弹出的对话框中单击“确定”按钮，如图 4-8 所示，让系统在网站根目录下生成“Web.config”文件。

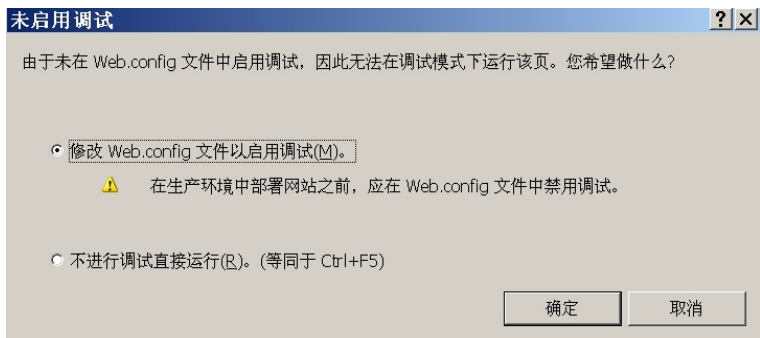


图 4-8 开启调试功能

③ 关闭弹出的浏览器窗口，在右侧的“解决方案资源管理器”中右键单击“Default.aspx”，在快捷菜单中选择“删除”，如图 4-9 所示，删除该页面。

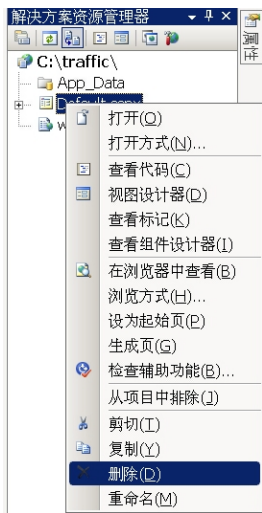


图 4-9 删除默认页

④ 在“解决方案资源管理器”窗口中，右键单击“traffic”项目，选择“新建文件夹”，如图 4-10 所示。将新文件夹命名为“web”，该文件夹主要用来保存“商业网站流量分析系统”的所有.aspx 文件。

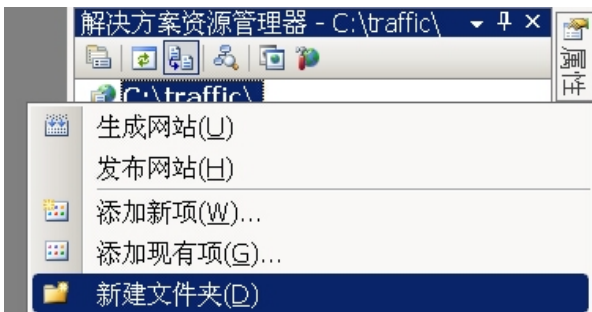


图 4-10 新建文件夹

⑤ 为了使通用网站流量分析系统具备统一的操作界面，使用母版页技术来统一网站风格。只有文字的网站，常常会使人感到厌倦，适当地在页面中添加一些图片可以起到美化站点的作用。在“traffic”项目下的“web”文件夹上用鼠标右击，选择“添加新项”命令，如图 4-11 所示。



图 4-11 在“web”文件夹下添加新项

在打开的“添加新项”窗口中选择“母版页”，如图 4-12 所示，单击“添加”按钮。

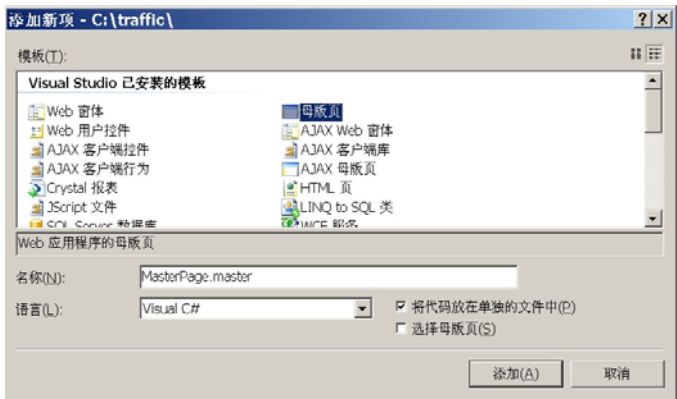


图 4-12 选择“母版页”

⑥ 右键单击“img”文件夹，选择“添加现有项”，如图 4-13 所示。弹出“添加现有项”对话框，选择待添加的图片文件，单击“添加”按钮，如图 4-14 所示。



图 4-13 添加现有项

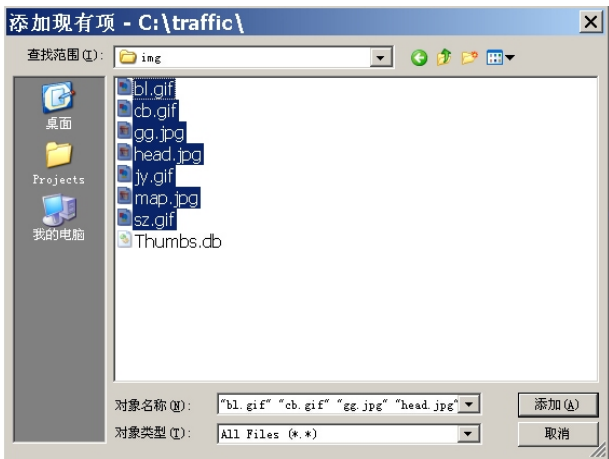


图 4-14 添加图片文件

⑦ 选择“源代码”模式，将以下代码添加到对应位置：

```
<head id="Head1" runat="server"><title>无标题页</title>
<asp:ContentPlaceHolder id="head" runat="server"></asp:ContentPlaceHolder>
<style type="text/css">
    .style1 { width: 230px; }
```

</head>

⑧ 在<body></body>标签对之间添加如下代码。

[illegible]

```

                                <span><a
href="http://www.sznet110.gov.cn/netalarm/index.jsp">
    深圳网络警
    <br />
    察报警平台 </a></span>
</p>
</td>
<td>
    <p>
        &nbsp;<asp:Image ID="Image3" runat="server"
            ImageUrl="~/img/gg.jpg" />
        <span><a
href="http://www.sznet110.gov.cn/webrecord/innernet/Welcome.jsp?bano=4403101010155">
            公共信息安全<br />
            全网网络监察</a></span>
    </p>
</td>
<td>
    <p>
        &nbsp;<asp:Image ID="Image4" runat="server"
            ImageUrl="~/img/jy.gif" />
        <span><a
href="http://www.hd315.gov.cn/beian/view.asp?bianhao=0272000112400002"
class="lcblack" target="_blank">经营性网站<br />
        备案信息</a></span>
    </p>
</td>
<td>
    <p>
        &nbsp;<asp:Image ID="Image5" runat="server"
            ImageUrl="~/img/bl.gif" />
        <span style="width: 64px;"><a
href="http://net.china.cn/chinese/index.htm"
class="lcblack">不良信息<br>
        举报中心</a></span>
    </p>
</td>
<td>
    <p>
        &nbsp;<asp:Image ID="Image6" runat="server"
            ImageUrl="~/img/cb.gif" />
        <span style="width: 64px;"><a
href="http://www.wenming.cn"
class="lcblack">中国文明网
        <br>
        传播文明 </a></span>
    </p>

```

```

</td>
</tr>
</table>
</div>
</td>
</tr>
</table>
</form>
</body>

```

⑨ 切换到“设计”视图模式，如图 4-15 所示。将“工具箱”\“导航”\“TreeView”控件拖到图 4-15 中的“【1】”处，如图 4-16 所示。



图 4-15 “设计”视图模式下的效果图



图 4-16 拖动控件

母版页效果如图 4-17 所示。



图 4-17 母版页效果图

⑩ 依次在“web”文件夹下添加 4 个窗体，分别命名为“Default.aspx”、“site.aspx”、“func.aspx”、“ip.aspx”；在创建的同时选中“选择母版页”，如图 4-18 所示。选择刚刚建立的母版页“MasterPage.master”，如图 4-19 所示。



图 4-18 添加窗体“Default.aspx”

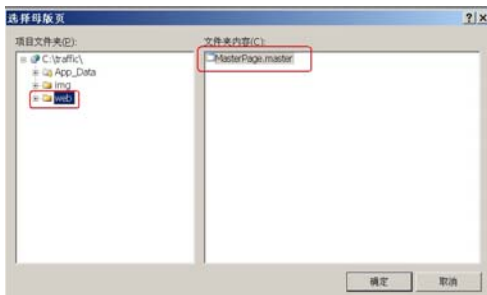


图 4-19 选择“web”文件夹下的“MasterPage.master”母版页

⑪ 在“解决方案资源管理器”的“traffic”项目上单击鼠标右键，在弹出的快捷菜单中选择“添加新项”，如图 4-20 所示。

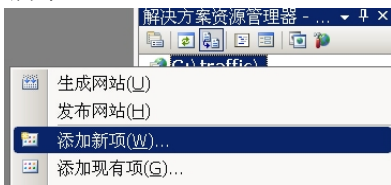


图 4-20 添加新项

⑫ 在打开的“添加新项”窗口中选择“站点地图”，如图 4-21 所示。



图 4-21 添加站点地图

⑬ 编辑 “Web.sitemap” 站点地图文件代码:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="web/default.aspx" title="首页" description="首页">
    <siteMapNode
url="web/site.aspx" title="网站流量" description="网站流量" />
    <siteMapNode url="web/func.aspx" title="栏目流量" description="栏目流量" />
    <siteMapNode url="web/ip.aspx" title="IP 统计" description="IP 统计" />
  </siteMapNode></siteMap>
```

⑭ 进入母版页 MasterPage.master 的 “设计” 模式，单击 “TreeView1” 控件的智能标记，如图 4-22 所示。

⑮ 在打开菜单中选择 “新建数据源”，如图 4-23 所示。



图 4-22 单击智能标记

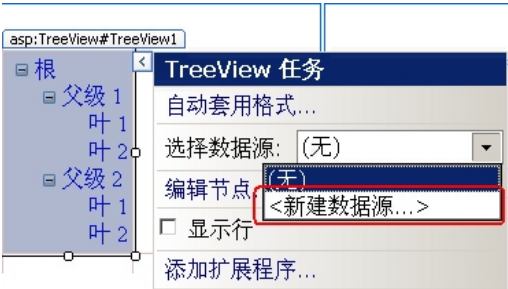


图 4-23 建立数据源

⑯ 在打开的 “数据源配置向导” 中选择 “站点地图”，如图 4-24 所示。

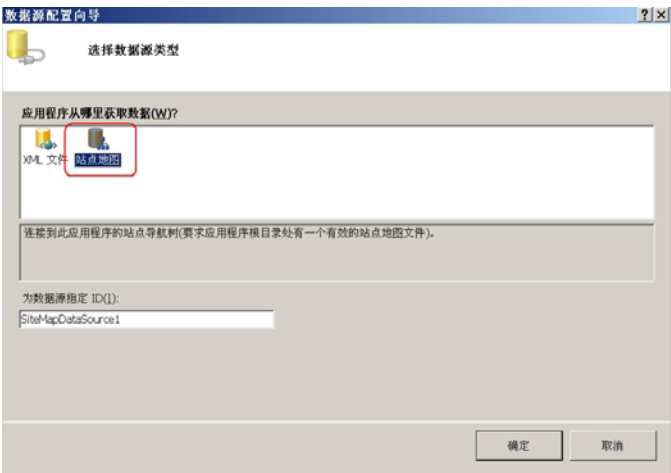


图 4-24 指定数据源类型

⑰ 单击 “TreeView1” 控件的智能标记后，在 “选择数据源” 一栏的下拉列表框中选择 “SiteMapDataSource1”，如图 4-25 所示。

⑱ 在 “解决方案资源管理器” 中设置 “Default.aspx” 页为起始页，如图 4-26 所示。

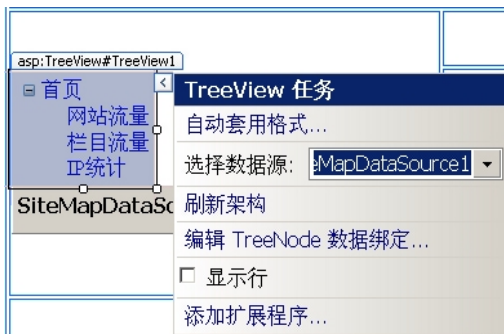


图 4-25 选择数据源

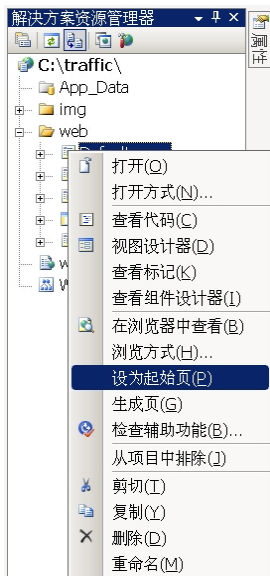


图 4-26 设置起始页

⑲ 按“F5”键查看网站运行效果。至此，网站母版页设计完毕，如图 4-27 所示。



图 4-27 母版页

4.1.6 设计显示当前日期和时间的Web自定义控件DisplayTime.ascx

程序开发步骤如下：


① 首先来预览网站流量功能页的运行效果，如图 4-28 所示。

② 商业网站流量分析系统主要提供 3 个功能：网站流量、栏目流量、IP 统计。在实现这 3 种功能的同时，还应该在页面上提示当前日期、当月访问量、当日访问量等信息。为了体现面向对象技术编程的代码可重用性，使用 Web 用户控件来设计以上 3 种信息的显示功能，即显示当前日期、当月访问量、当日访问量。




图 4-28 网站流量功能页运行效果

③ 首先在网站根目录新建文件夹“UCtrls”，添加新项“Web 用户控件”，并且命名为“DisplayTime.ascx”，该 Web 用户控件的功能为显示当前日期和时间。

④ 在“DisplayTime.ascx”页面的  “源代码”视图添加以下代码：

```
<asp:Label ID="Label1" runat="server"></asp:Label>
```

⑤ 然后单击“DisplayTime.ascx”页面前面的“+”，即  **DisplayTime.ascx**。

⑥ 再双击“DisplayTime.ascx.cs”，添加以下代码，最后保存。

```
public partial class UCtrls_DisplayTime : System.Web.UI.UserControl{
    protected void Page_Load(object sender, EventArgs e){
        DateTime d = DateTime.Now;
        Label1.Text = d.ToString ();
    }
}
```

4.1.7 设计数据访问层的数据访问类 DataBase.cs

前面提到的显示当月访问量和当日访问量的功能主要是通过访问 traffic 数据库 Func 表的数据来实现的，为此需要为网站设计数据访问层的操作类 DataBase.cs。

① 在网站中打开“添加新项”窗口，选择“类”，如图 4-29 所示，单击“添加”按钮，在打开窗口中单击“是”按钮，如图 4-30 所示。其目的是让 VS2008 开发平台（以下简称 IDE）自动为网站产生 App_Code 文件夹，进而更好地管理“.cs”类文件。

② 删除刚刚建立的“Class1.cs”文件，并在“App_Code”文件夹下建立“DAL”文件夹。再次打开“添加新项”窗口，选择“类”，命名为“DataBase.cs”，如图 4-31 所示。



图 4-29 添加类

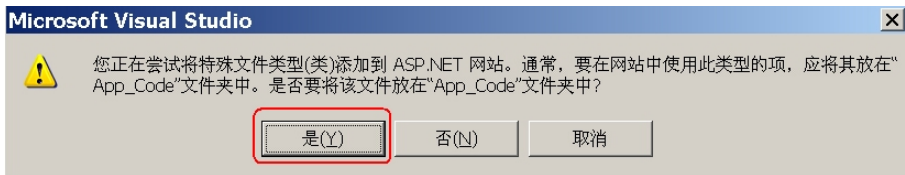


图 4-30 单击“是”按钮

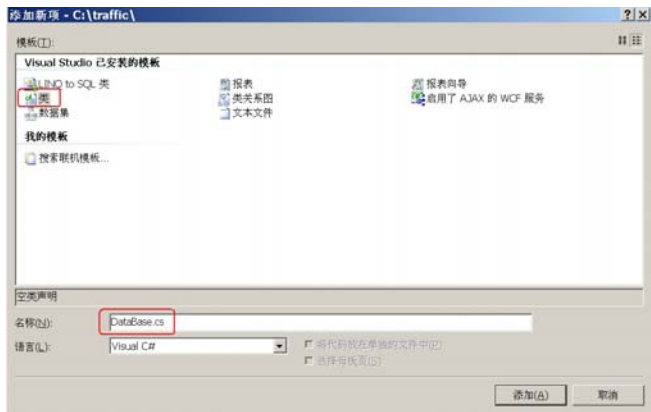


图 4-31 命名类

③ 在“DataBase.cs”中添加以下代码：

```
namespace traffic.DAL{
    public class Database : IDisposable{
        protected SqlConnection Connection;
        private String ConnectionString;
        public Database (){
            //获取 web.config 里保存的名为 conn 的数据库连接字符串
            ConnectionString = ConfigurationManager.AppSettings["conn"];
        }
        //当连接对象资源被系统回收时，调用关闭连接方法
        ~Database (){
            try{
```

```

        if (Connection != null)
            Connection.Close();
    }
    catch (Exception e){
    }
    try{
        Dispose();
    }
    catch { }
}
//打开数据库连接
protected void Open (){
    if (Connection == null){
        try{
            Connection = new SqlConnection (ConnectionString);
        }
        catch (Exception e){}
    }
    if (Connection.State.Equals (ConnectionState.Closed) ){
        try{
            Connection.Open();
        }
        catch (Exception e){}
    }
}
//关闭数据库连接
public void Close(){
    try{
        if (Connection != null)
            Connection.Close();
    }
    catch (Exception e){}
}
//释放连接对象 Connection 所占用的系统资源
public void Dispose (){
    //确保连接被关闭
    try{
        if (Connection != null){
            Connection.Dispose();
            Connection = null;
        }
    }
    catch (Exception e){
    }
}
//返回 DataReader 数据集
public SqlDataReader GetDataReader(String SqlString){

```

```

        Open();
    try{
        SqlCommand cmd = new SqlCommand(SqlString, Connection);
        return cmd.ExecuteReader(System.Data.CommandBehavior.CloseConnection);
    }
    catch (Exception e){
        return null;
    }
}
}
}

```

④ 在代码中有一个字符串变量 **ConnectionString** 用来保存数据库连接字符串，而它的内容来源于网站根目录下的“web.config”文件。为此，还需要在该文件的指定位置添加以下代码，其中“XX”为网站程序所在的服务器机器名称。

```

<configuration><appSettings>
    <add key="conn" value="Data Source=XX\SQLEXPRESS;Initial Catalog=traffic;Integrated
Security=True"/></appSettings>
</configuration>

```

4.1.8 设计显示当月访问量的Web用户控件DisplayMonthCount.ascx

程序开发步骤如下：

- ① 在网站项目文件夹“UCtrls”中添加新项“Web 用户控件”，命名为“DisplayMonthCount.ascx”。
- ② 进入“DisplayMonthCount.ascx”页面的代码视图模式，添加如下代码：

```

<asp:Label ID="Label1" runat="server"></asp:Label>

```

③ 进入“DisplayMonthCount.ascx.cs”页面，添加以下代码，其中“using traffic.DAL”；为引入刚刚建立的 DataBase.cs 数据范围类所在的命名空间。通过创建 DateTime 对象获取当前月份，并将当前月份作为检索 Func 表的条件获取满足该月份的记录总数。即 select count(*) from func where month(loadtime)=month，其中 count(*)为自动求和函数。值得注意的是，当前访问表 Func 的方式为通过构建 SQL 语句在数据库以外的环境下操作，需要对数据库查询语句进行字符串式的封装构建，因此就有了定义一个 String 类型的数据库查询字符串 str 的定义，在此读者应重点留意带有查询条件（参数）的查询字符串 String str=@"select count(*) from func where month(loadtime)=" + month; 的构建特点。最后将查询结果以一个 DataReader 数据集的方式返回，并以字符串的方式赋值给标签控件 Label1 的 Text 值。

```

using traffic.DAL;
public partial class UCtrls_DisplayMonthCount : System.Web.UI.UserControl{
    protected void Page_Load (object sender, EventArgs e){
        DateTime today=DateTime.Now;
        int month=today.Month;
        String str=@"select count (*) from func where month (loadtime)=" + month;
        Database database = new Database ();
    }
}

```

```

        SqlDataReader sdr = database.GetDataReader (str);
        if (sdr.HasRows){
            while (sdr.Read ())
            {
                Label1.Text = sdr[0].ToString ();
            }
        }
    }
}

```

4.1.9 设计显示当日访问量Web用户控件getDayCount.ascx

程序开发步骤如下：

① 在网站项目的文件夹“UCtrl”中添加新项“Web 用户控件”，命名为“getDayCount.ascx”。进入“DisplayMonthCount.ascx”页面的代码视图模式，添加如下代码：

```
<asp:Label ID="Label1" runat="server"></asp:Label>
```

② 进入“DisplayMonthCount.ascx.cs”页面添加如下代码。当日访问量实现思想与前面的“显示当月访问量”设计思路相同，只是筛选表 Func 记录的条件改成了当前日期，即 DateTime.Now.Day。

```

protected void Page_Load (object sender, EventArgs e){
    DateTime today = DateTime.Now;
    int day = today.Day;
    String str = @"select count(*) from func where day (loadtime)="+ day;
    Database database = new Database ();
    SqlDataReader sdr = database.GetDataReader (str);
    if (sdr.HasRows){
        while (sdr.Read ())
        {
            Label1.Text = sdr[0].ToString ();
        }
    }
}
}

```

4.2 系统实现

4.2.1 site站点流量统计

商业网站流量分析统计系统主要提供 3 个功能：网站流量、栏目流量、IP 统计。这 3 个功能实现的前提条件是用户要先设定查询条件：查询年份、时间跨度等。并且从效果图上可以看出，网站流量统计功能的实现是以用户选择的查询年份为依据的，因此对于网站流量分析功能的实现主要从以下几个方面入手。

程序开发步骤如下：

① 在 site.aspx 的“源代码”模式下添加以下代码，产生一个多行一列的表格：

```

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"Runat="Server">
<table id="table1" style="width:100%;"><tr><td>今天是: </td> </tr>
<tr><td>本月访问量: </td> </tr>
<tr><td>当日访问量: </td> </tr></table>
</asp:Content>

```

② 将前面创建的 3 个 Web 用户控件图标从右侧的“解决方案资源管理器”拖曳到 site.aspx 页面的对应区域。并在 site.aspx 的“源代码”模式下加以调整，使得一个 Web 控件占用一行，此时效果如图 4-32 所示。



图 4-32 site.aspx 添加自定义控件的效果

代码如下：

```

<tr><td>今天是: <uc1:DisplayTime ID="DisplayTime1" runat="server" /></td></tr>
<tr><td>本月访问量: <uc2:DisplayMonthCount ID="DisplayMonthCount1" runat="server" /></td>
</tr>
<tr><td>当日访问量: <uc3:getDayCount ID="getDayCount1" runat="server" /></td></tr>

```

③ 为页面添加用于设定查询年份的下拉列表框。在 site.aspx 的“源代码”模式下添加以下代码：

```

<tr><td>选择: <asp:DropDownList ID="DropDownList1" runat="server">
</asp:DropDownList>年<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="
查询" Width="99px" /></td></tr>

```

④ 在 site.aspx.cs 下添加用于初始化下拉列表框 DropDownList1 的代码：

```

protected void Page_Load(object sender, EventArgs e){
    if (!IsPostBack){
        //初始化 (年) 下拉列表框 2000~2099
        for (int i = 2099; i > 2000; i--){
            DropDownList1.Items.Insert(0, i.ToString());
        }
    }
}

```

⑤ 当用户选择一个年份，并单击“查询”按钮时应该显示符合该年份的所有网站流量记录信息。由于数据来源于服务器上的数据库，若此时选择的查询年份恰巧没有访问记录，则查询后不会有任何记录显示在页面上，此种现象和查询等待时的表现相同，常会给人以查询效率低下、系统响应迟钝的错觉。因此，建议增加一个用于显示查询结果的提示标签，当查询到符合年份的数据时，显示查询到某某年份如下数据，而没有查询到符合该年份的数据时则提示该年份尚未存在浏览记录的字样，使网站的功能更趋于人性化。在 site.aspx 的“源代码”模式下添加以下代码：

```
<tr><td style="text-align:center"><asp:Label ID="lblmeg" runat="server" Text=""></asp:Label>
</td> </tr>
```

⑥ 进入 site.aspx 的“设计”模式，应该产生以下效果，如图 4-33 所示。



图 4-33 site.aspx “设计”模式的效果图

⑦ 单击“查询”按钮，使 IDE 自动生成“查询”按钮的单击处理事件。重新切换到 site.aspx 的“源代码”模式下，添加如下代码：

```
<tr><td style="text-align:left"><asp:GridView ID="GridView1" runat="server" CellPadding="4"
ForeColor="#333333" PageSize="1" Font-Bold="False" Width="770px"
DataSourceID="SqlDataSource1" AutoGenerateColumns="False">
<FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
<RowStyle BackColor="#EFF3FB" BorderStyle="Dotted" />
<Columns>
<asp:BoundField DataField="年度" HeaderText="年度"/>
<asp:TemplateField HeaderText="访问次数">
<ItemTemplate>
<asp:Image ID="siteImage" runat="server" Height="5px"
ImageUrl="~/img/map.jpg"
Width='<%# Int32.Parse(Eval ( "访问次数").ToString()) %>' />
<%# Int32.Parse ( Eval ( "访问次数").ToString ( ) ) %>次
</ItemTemplate>
</asp:TemplateField>
</Columns>
<HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White"
HorizontalAlign="left" />
<AlternatingRowStyle BackColor="White" />
</asp:GridView>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%= $ ConnectionStrings:trafficConnectionString2 %>"
SelectCommand="SELECT YEAR ( LoadTime ) AS '年度', COUNT ( * ) AS '访问次数'
FROM Func
WHERE ( YEAR ( LoadTime ) = @year )
GROUP BY YEAR ( LoadTime )">
```

```

<SelectParameters>
    <asp:ControlParameter ControlID="DropDownList1" Name="year"
        PropertyName="SelectedValue" />
</SelectParameters>
</asp:SqlDataSource></td></tr>

```

⑧ 以上代码用到了 web.config 文件夹里的数据库连接字符串 trafficConnectionString2，因此打开 web.config，找到<connectionStrings/>并将它替换成以下代码，其中的 Data Source=lh\sqlexpress 表示 lh 服务器的 sqlexpress 服务：

```

<connectionStrings>
    <add name="trafficConnectionString" connectionString="Data Source=LH\SQLEXPRESS;Initial
Catalog=traffic;Integrated Security=True"
        providerName="System.Data.SqlClient" />
    <add name="trafficConnectionString2" connectionString="Data Source=lh\sqlexpress;Initial
Catalog=traffic;Integrated Security=True"
        providerName="System.Data.SqlClient" />

```

⑨ 进入 site.aspx.cs，在“查询”按钮的单击处理事件中添加如下代码：

```

string year = DropDownList1.SelectedValue;
GridView1.DataSourceID = SqlDataSource1.ID;
GridView1.DataBind ();
if ( GridView1.Rows.Count== 0 ){
    lblmeg.Text = year + "年度无访问量！";
}
else{
    lblmeg.Text = year + "年度访问量";
}

```

⑩ 设置 site.aspx 为起始页，按“F5”键，选择年份，单击“查询”按钮，测试查询效果。当选择年份有访问记录时，演示效果如图 4-34 所示；当选择年份没有访问记录时，演示效果如图 4-35 所示。



图 4-34 当选择年份有访问记录时的演示效果图



图 4-35 当选择年份没有访问记录时的演示效果图

4.2.2 func栏目流量统计

本项目根据指定年月日区间的栏目流量进行统计，并以图形和数值的方式予以呈现，整个代码编写过程同前面的站点流量统计相似。

程序开发步骤如下：

① 进入 func.aspx 的“源代码”模式，添加以下代码：

```
<% @ Register src="./UControls/DisplayTime.ascx" tagname="DisplayTime" tagprefix="uc1" %>
<% @ Registersrc="./UControls/DisplayMonthCount.ascx" tagname="DisplayMonthCount"tagprefix="uc2"%>
<% @ Register src="./UControls/getDayCount.ascx" tagname="getDayCount" tagprefix="uc3" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<table id="table1" style="width:100%;">
<tr><td> 今天是: <uc1:DisplayTime ID="DisplayTime1" runat="server" /></td> </tr>
<tr><td>本月访问量: <uc2:DisplayMonthCount ID="DisplayMonthCount1" runat="server" /></td>
</tr>
<tr><td>当日访问量: <uc3:getDayCount ID="getDayCount1" runat="server" />
</td></tr></table></asp:Content>
```

② 栏目流量统计功能实现的前提条件是根据指定的时间段进行查询，因此需要给出用于选择时间段的控件。这里使用下拉列表框来完成该功能，添加如下代码：

```
<tr><td>从: <asp:DropDownList ID="DropDownList1" runat="server"></asp:DropDownList>
年<asp:DropDownList ID="DropDownList2" runat="server"></asp:DropDownList>
月<asp:DropDownList ID="DropDownList3" runat="server"></asp:DropDownList>
日</td></tr>
<tr><td>到: <asp:DropDownList ID="DropDownList7" runat="server"></asp:DropDownList>
年<asp:DropDownList ID="DropDownList8" runat="server"></asp:DropDownList>
月<asp:DropDownList ID="DropDownList9" runat="server"></asp:DropDownList>
日<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="查询"
Width="106px" /></td></tr>
```


③ 在 func.aspx.cs 的 Page_Load 方法内添加如下代码，用来初始化设置时间段下拉列表框的数据：

```
if (!IsPostBack) {
    int i;
    //初始化(年)下拉列表框 2008~2009
    for (i = 2009; i > 2007; i--){
        DropDownList1.Items.Insert ( 0, i.ToString () );
        DropDownList7.Items.Insert ( 0, i.ToString () );
    }
    //初始化(月)下拉列表框 1~12
    for (i = 12; i > 0; i--){
        DropDownList2.Items.Insert ( 0, i.ToString () );
        DropDownList8.Items.Insert ( 0, i.ToString () );
    }
    //初始化(日)下拉列表框 1~31
    for (i = 31; i > 0; i--){
        DropDownList3.Items.Insert ( 0, i.ToString () );
        DropDownList9.Items.Insert ( 0, i.ToString () );
    }
}
```

④ 添加信息提示标签：

```
<tr><td style="text-align:center"><asp:Label ID="lblmeg" runat="server" Text=""></asp:Label>
</td></tr>
```

⑤ 添加显示查询结果的 GridView 和数据源控件 SqlDataSource：

```
<tr><td><table id="table3" style="width:100%;">
    <tr><td > <asp:GridView ID="GridView1" runat="server" CellPadding="4"
ForeColor="#333333" PageSize="30" Font-Bold="False" Width="770px" AllowPaging="False"
AutoGenerateColumns="False" DataSourceID="SqlDataSource1" >
    <FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
    <RowStyle BackColor="#EFF3FB" BorderStyle="Dotted" />
    <Columns><asp:BoundField DataField="栏目名称"HeaderText="栏目名称"SortExpression="栏目名称"
/><asp:TemplateField HeaderText="访问次数">
        <ItemTemplate><asp:Image ID="siteImage" runat="server" Height="5px"
ImageUrl="~/img/map.jpg" Width="<%= Int32.Parse (Eval ("访问次数").ToString ()) %>"/><%=
Int32.Parse (Eval ("访问次数").ToString ()) %>次
        </ItemTemplate> </asp:TemplateField></Columns>
        <HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White"
HorizontalAlign="left" />
        <AlternatingRowStyle BackColor="White" />
    </asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%= $ ConnectionStrings:trafficConnectionString %>"
        SelectCommand="select funcname as 栏目名称, count (*) as 访问次数 from
func where loadtime>=@sj1 and loadtime<=@sj2 group by funcname">
```

```

<SelectParameters>
    <asp:Parameter Name="sj1" />
    <asp:Parameter Name="sj2" />
</SelectParameters>
</asp:SqlDataSource>
</td></tr></table></td></tr>

```

⑥ 切换到 func.aspx 的“设计”模式，单击“查询”按钮，在 func.aspx.cs 的按钮单击处理事件 Button1_Click 中添加如下代码：

```

sj1 = DropDownList1.SelectedValue + "-" + DropDownList2.SelectedValue + "-" +
DropDownList3.SelectedValue;
sj2 = DropDownList7.SelectedValue + "-" + DropDownList8.SelectedValue + "-" +
DropDownList9.SelectedValue;
SqlDataSource1.SelectParameters[0].DefaultValue = sj1;
SqlDataSource1.SelectParameters[1].DefaultValue = sj2;
GridView1.DataSourceID = SqlDataSource1.ID;
GridView1.DataBind ();
if (GridView1.Rows.Count == 0 ){
    lblmeg.Text = sj1 + "至" + sj2 + "栏目无访问量！";
}else{lblmeg.Text = sj1 + "至" + sj2 + "栏目访问量";}

```

⑦ 其中用到了两个字符串变量 sj1 和 sj2，需要现行定义为全局变量，在所有的方法体外定义：

```

public string sj1;
public string sj2;

```

⑧ 将 func.aspx 设定为起始页，按“F5”键测试查询效果，如图 4-36 所示。



图 4-36 测试查询效果

⑨ 设定起始年份为 2008，终止年份为 2016，单击“查询”按钮，查询栏目访问量，如图 4-37 所示。



图 4-37 栏目访问量查询结果

4.2.3 IP流量统计

本项目根据指定年月日区间的 IP 流量进行统计，并以图形和数值的方式予以呈现，整个代码编写过程同前面的站点流量统计相似。

程序开发步骤如下：

① 进入 ip.aspx 的“源代码”模式，在页面第二行处添加以下代码，实现 Web 用户控件的正确引用：

```
<% @ Register src="..\UCtrls/DisplayTime.ascx" tagname="DisplayTime" tagprefix="uc1" %>
<% @ Register src="..\UCtrls/DisplayMonthCount.ascx" tagname="DisplayMonthCount" tagprefix="uc2"%>
<% @ Register src="..\UCtrls/getDayCount.ascx" tagname="getDayCount" tagprefix="uc3" %>
```

② 继续在 Content2 标签对之间添加以下代码，完善界面要素：

```
<asp:Content ID="Content2" runat="server" contentplaceholderid="ContentPlaceHolder1">
    <table id="table1" style="width:100%;"><tr><td>
        今天是: <uc1:DisplayTime ID="DsplayTime1" runat="server" /></td></tr>
        <tr><td>本月访问量: <uc2:DisplayMonthCount ID="DisplayMonthCount1" runat="server" />
        </td></tr>
        <tr><td>当日访问量: <uc3:getDayCount ID="getDayCount1" runat="server" /></td></tr>
        <tr><td>从: <asp:DropDownList ID="DropDownList1" runat="server"></asp:DropDownList>
        年<asp:DropDownList ID="DropDownList2" runat="server"></asp:DropDownList>
        月<asp:DropDownList ID="DropDownList3" runat="server"></asp:DropDownList>
        日</td></tr><tr>
        <td>到: <asp:DropDownList ID="DropDownList7" runat="server"></asp:DropDownList>
        年<asp:DropDownList ID="DropDownList8" runat="server"></asp:DropDownList>
        月<asp:DropDownList ID="DropDownList9" runat="server"></asp:DropDownList>
        日<asp:Button ID="Button2" runat="server" onclick="Button1_Click" Text="查询"
        Width="99px" /></td></tr><tr>
        <td style="text-align:center"><asp:Label ID="lblmeg" runat="server"
        Text=""></asp:Label></td></tr></tr>
```

③ 进入 ip.aspx.cs, 在 Page_Load 方法中添加以下代码, 初始化时间段选择控件的数据:

```
if (!IsPostBack){
    int i;
    //初始化(年)下拉列表框 2008~2099
    for (i = 2099; i > 2007; i -){
        DropDownList1.Items.Insert (0, i.ToString ());
        DropDownList7.Items.Insert (0, i.ToString ());
    }
    //初始化(月)下拉列表框 1~12
    for (i = 12; i > 0; i -){
        DropDownList2.Items.Insert (0, i.ToString ());
        DropDownList8.Items.Insert (0, i.ToString ());
    }
    //初始化(日)下拉列表框 1~31
    for (i = 31; i > 0; i -){
        DropDownList3.Items.Insert (0, i.ToString ());
        DropDownList9.Items.Insert (0, i.ToString ());
    }
}
```

④ 进入 ip.aspx 的“源代码”模式, 添加用于显示统计数据的 GridView 和 SqlDataSource 控件的操作代码:

```
<tr><td><table id="table3" style="width:100%;"><tr><td>
    <asp:GridView ID="GridView1" runat="server" CellPadding="4"
ForeColor="#333333" PageSize="30" Font-Bold="False"
    Width="770px"
    AllowPaging="True"
    AutoGenerateColumns="False"
    DataSourceID="SqlDataSource1" >
    <PagerSettings FirstPageText="" LastPageText="" NextPageText="" Position="TopAndBottom"
PreviousPageText="" PageButtonCount="1" Visible="False" />
    <FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
    <RowStyle BackColor="#EFF3FB" BorderStyle="Dotted" />
</Columns><asp:BoundField DataField="ip" HeaderText="IP 地址" SortExpression="ip" />
    <asp:TemplateField HeaderText="访问次数">
        <ItemTemplate>
            <asp:Image ID="siteImage" runat="server" Height="5px"
ImageUrl="~/img/map.jpg"
            Width='<%# Int32.Parse(Eval("访问次数").ToString ()) %>' /><%#
Int32.Parse (Eval ("访问次数").ToString ()) %>次
        </ItemTemplate>
    </asp:TemplateField>
</Columns>
    <HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White"
HorizontalAlign="left" />
    <AlternatingRowStyle BackColor="White" />
```

```

        </asp:GridView>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%%$ ConnectionStrings:trafficConnectionString %>"
            SelectCommand="select ip, count(*) as 访问次数 from func where
loadtime>=@sj1 and loadtime<=@sj2 group by ip">
            <SelectParameters>
                <asp:Parameter Name="sj1" />
                <asp:Parameter Name="sj2" />
            </SelectParameters>
        </asp:SqlDataSource>
    </td>
</tr>
</table>
</td>
</tr>
</table>
</asp:Content>

```

⑤ 在 ip.aspx 的“设计”模式单击“查询”按钮，在单击处理方法 Button1_Click 中添加以下代码：

```

sj1 = DropDownList1.SelectedValue + "-" + DropDownList2.SelectedValue + "-" +
DropDownList3.SelectedValue;
sj2 = DropDownList7.SelectedValue + "-" + DropDownList8.SelectedValue + "-" +
DropDownList9.SelectedValue;
SqlDataSource1.SelectParameters[0].DefaultValue = sj1;
SqlDataSource1.SelectParameters[1].DefaultValue = sj2;
GridView1.DataSourceID = SqlDataSource1.ID;
GridView1.DataBind ();
if (GridView1.Rows.Count == 0){
    lblmeg.Text = sj1 + "至" + sj2 + "无访问量! ";}
else{lblmeg.Text = sj1 + "至" + sj2 + "IP 访问量";
}

```

⑥ 同样，需要定义全局字符串变量 sj1 和 sj2：

```

public string sj1;
public string sj2;

```

⑦ 将 ip.aspx 设定为起始页，按“F5”键，设定查询时间段，单击“查询”按钮，观察 IP 统计功能执行效果，如图 4-38 所示。



图 4-38 IP 统计功能执行效果

4.2.4 相关技能与要点

以上 3 个功能站点流量统计、栏目流量统计、IP 流量统计的设计思想基本一致，都是根据指定年份或时间段查询符合条件的记录，并通过分组方式统计记录数量，再以该数量作为图例的显示宽度，页面效果美观、大方，而实际代码的编写难度却较低，有利于初学者掌握。

4.2.5 任务小结

本章涉及的技术比较简单也易于实现，虽然叙述篇幅较长，但实际程序源代码在 4 个章节的项目中是最短的。因此，本部分内容较为适合对开发工具功能不十分熟悉、对开发过程不太了解的初学者首先掌握。

4.2.6 练习题

针对以上所学内容，设计以年为时间跨度的 SQL 语句。

参 考 文 献

- [1] Mike Docherty. 面向对象分析与设计 (UML 2.0 版). 北京: 清华大学出版社, 2006.4
- [2] 陈轮、刘蕾. ASP.NET 3.5 网络数据库开发实例自学手册. 北京: 电子工业出版社, 2008.5
- [3] 张领. ASP.NET 项目开发全程实录. 北京: 清华大学出版社, 2008.6
- [4] 明日科技. Visual C#开发技术大全. 北京: 人民邮电出版社, 2007.11